# Traitement d'image

(compléments d'informations à destination du professeur)

Raoul HATTERER

8 février 2019

## Table des matières



# 1 Codage RVB et niveau de gris

— Aller sur colors RGB et tester ce que l'on obtient si l'on remplace chacune des valeurs R, V et B d'un pixel par la moyenne des sous-pixels.
— Essayer pour plusieurs couleurs.

# 2 Images de départ

Le professeur propose une ou plusieurs images couleurs de départ.

# 3 Comment lire un pixel

## 3.1 Installation de PIL

À faire au préalable par le professeur.

```
pip3 install pillow
```

FIGURE 1 – Image de départ (Pomme Linux)

FIGURE 2 – Image de départ (Colin de Californie)

### 3.2 Activité

Après avoir fait quelques recherches sur ce qu'est un "pixel", voyons comment lire le pixel de coordonnées (100,250).

```
from PIL import Image
img = Image.open("pomme.jpg")
r,v,b=img.getpixel((100,250))
print("canal rouge : ",r,"canal vert : ",v,"canal bleu : ",b)
```

```
('canal rouge : ', 19, 'canal vert : ', 88, 'canal bleu : ', 192)
```

## 4 Comment écrire un pixel

### 4.1 Code

```
from PIL import Image
img = Image.open("pomme.jpg")
img.putpixel((5,5),(255,0,0))
img.show()
```

### 4.2 Question

Identifier où se trouve l'origine de l'image.

### 4.3 Réponse

Les élèves écrivent un pixel de couleur spécifique à la position (0,0) ou à proximité et cherchent en examinant l'image, près de quel coin il apparaît. On identifie ainsi que l'origine est en haut à gauche.

## 5 Que fait le programme suivant ?

```
# coding: utf-8                                  # Spécifie l'encodage (ici unicode) du code source
from PIL import Image                            # Importation de la librairie PILLOW (gestion image)
img = Image.open('pomme.jpg')                    # Mise en memoire dans la variable 'img' du fichier
#                                                # pomme.jpg qui doit être dans le même répertoire que
#                                                # le programme
largeur_image,hauteur_image=img.size             # Python autorise les affectations multiples.
#                                                # img.size est un attribut (une variable intrinsèque
#                                                # à la variable img) avec les dimensions de l'image
#                                                # sous forme de tupple (= liste non modifiable).

for y in range(hauteur_image):                   # Boucle pour parcourir les toutes les lignes
    for x in range(largeur_image):               # Boucle imbriquée pour parcourir les pixels de la
        #                                        # ligne en cours
        rouge,vert,bleu=img.getpixel((x,y))      # Méthode getpixels() appliquée à la variable img qui
        #                                        # renvoie les valeurs R,V,B du pixel à la position x,y
        nouveau_rouge=vert                       # Le vert prend l'intensité du rouge
        nouveau_vert=bleu                        # Le bleu prend l'intensité du vert
        nouveau_bleu=rouge                       # Le rouge prend l'intensité du bleu
        img.putpixel((x,y),(nouveau_rouge,nouveau_vert,nouveau_bleu)) # Méthode putpixel()
#                                                # qui remplace les valeurs R, V, B du pixel en x,y

img.show()                                       # Affichage de l'image
img.save("pommeMystere.jpg")                     # Sauvegarde de l'image obtenue
print(img.size)                                  # Affichage du tupple avec la taille de l'image
```

```
(480, 300)
```

On analyse le code ci-dessus (sans forcément rentrer dans les détails) qui servira de base pour le défi suivant.

FIGURE 3 – Résultat du programme mystère

Les couleurs ont été permutées.

## 6 Passage d'une image en niveau de gris (codé RVB sur 3 octets)

Après avoir fait quelques recherches sur les "images en niveaux de gris", écrivez un programme qui transforme une "image couleur" en une "image en niveaux de gris".

Petite astuce qui pourrait vous aider : en Python pour avoir une division entière (le résultat est un entier), il faut utiliser l'opérateur // à la place de l'opérateur /

Remarque : On donne l'algorithme aux élèves (ou on le construit avec eux) ; ils doivent alors programmer le passage d'une image couleur à une image en niveaux de gris.

```
from PIL import Image
img = Image.open("pomme.jpg")
largeur_image,hauteur_image=img.size

for y in range(hauteur_image):
    for x in range(largeur_image):
        rouge,vert,bleu=img.getpixel((x,y))
        nouveau_rouge=(vert+bleu+rouge)//3
        nouveau_vert=(vert+bleu+rouge)//3
        nouveau_bleu=(vert+bleu+rouge)//3
        img.putpixel((x,y),(nouveau_rouge,nouveau_vert,nouveau_bleu))

img.show()
img.save("pommegrise.jpg")
```

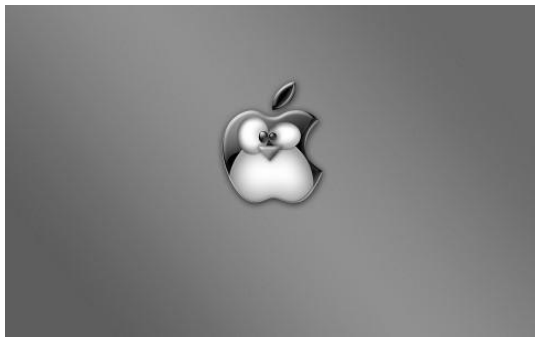

FIGURE 4 – Pomme Linux en niveaux de gris (codé RVB)

```
from PIL import Image
img = Image.open("California_Quail.jpg")
largeur_image,hauteur_image=img.size

for y in range(hauteur_image):
    for x in range(largeur_image):
        rouge,vert,bleu=img.getpixel((x,y))
        nouveau_rouge=(vert+bleu+rouge)//3
        nouveau_vert=(vert+bleu+rouge)//3
        nouveau_bleu=(vert+bleu+rouge)//3
        img.putpixel((x,y),(nouveau_rouge,nouveau_vert,nouveau_bleu))

img.show()
img.save("colingris.jpg")
```

FIGURE 5 – Colin de Californie en niveaux de gris RVB

# 7 Passage d'une image en niveau de gris (codé L sur 1 seul octet)

## 7.1 Utilisation du mode L (luminance) pour les images en nuances de gris

### 7.1.1 Pomme Linux

```
1 from PIL import Image
2 img = Image.open("pomme.jpg").convert("L")
3 img.show()
4 img.save("pommegriseL.jpg")
```

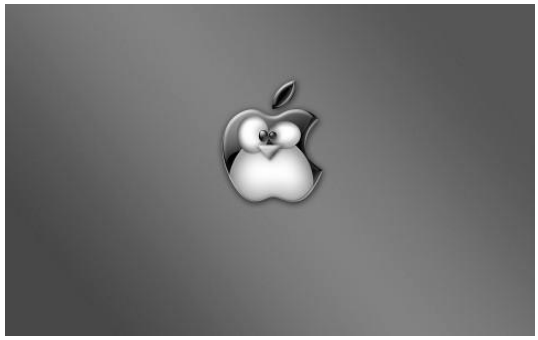

FIGURE 6 – Image en niveaux de gris (sans redondance)

### 7.1.2 Colin de Californie

```
1 from PIL import Image
2 img = Image.open("California_Quail.jpg").convert("L")
3 img.show()
4 img.save("colingrisL.jpg")
```

Comparer la taille des différents fichiers. Conclure.

*Réponse : codée avec un octet par pixel, l'image (les datas) prend moins de place donc le fichier est moins lourd (la compression jpeg atténue le phénomène).*

FIGURE 7 – Colin de Californie en niveaux de gris (luminance)

### 7.2 Existe-t-il d'autres modes ?

Les modes supportés par `Pillow` sont :
— `1` (1-bit pixels, black and white, stored with one pixel per byte)
— `L` (8-bit pixels, black and white)
— `P` (8-bit pixels, mapped to any other mode using a color palette)
— `RGB` (3x8-bit pixels, true color)
— `RGBA` (4x8-bit pixels, true color with transparency mask)
— `CMYK` (4x8-bit pixels, color separation)
— `YCbCr` (3x8-bit pixels, color video format)
— `LAB` (3x8-bit pixels, the L*a*b color space)
— `HSV` (3x8-bit pixels, Hue, Saturation, Value color space)
— `I` (32-bit signed integer pixels)
— `F` (32-bit floating point pixels)

## 8 Récréation ou challenge ?

### 8.1 Créer une image en négatif

```
from PIL import Image
img = Image.open("pomme.jpg")
largeur_image,hauteur_image=img.size

for y in range(hauteur_image):
    for x in range(largeur_image):
        rouge,vert,bleu=img.getpixel((x,y))
        nouveau_rouge=255-rouge
        nouveau_vert=255-vert
        nouveau_bleu=255-bleu
        img.putpixel((x,y),(nouveau_rouge,nouveau_vert,nouveau_bleu))

img.show()
img.save("pommeNegatif.jpg")
```

FIGURE 8 – Négatif

## 8.2 Diagonale

Créer le programme qui garde l'image d'origine au-dessus d'une diagonale et qui transforme en niveaux de gris en-dessous de celle-ci.

```
from PIL import Image
img = Image.open("pomme.jpg")
largeur_image,hauteur_image=img.size

for y in range(hauteur_image):
    tailleDiag=y*largeur_image//hauteur_image
    for x in range(tailleDiag):
        rouge,vert,bleu=img.getpixel((x,y))
        nouveau_rouge=(vert+bleu+rouge)//3
        nouveau_vert=(vert+bleu+rouge)//3
        nouveau_bleu=(vert+bleu+rouge)//3
        img.putpixel((x,y),(nouveau_rouge,nouveau_vert,nouveau_bleu))

img.show()
img.save("pommemisgrise.jpg")
```

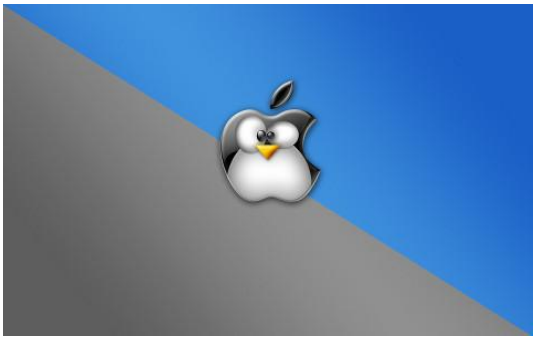

FIGURE 9 – Pomme coupée