

Photo numérique : traitement d'images

« Un bon traitement contre l'insomnie est de beaucoup dormir. »

W. C. Fields

I. Définition

Le traitement d'images est une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information. On peut citer par exemple : la création d'effets spéciaux au cinéma, l'amélioration de la qualité des photos en corrigeant les conditions de prise de vue, la détection de lettres ou de visages, reconnaissance d'objets, l'identification de zones cancéreuses en imagerie médicale, mesure, caractérisation du contenu d'images, la compression JPEG, analyse et estimation du mouvement dans les séquences d'images, etc.

Pour traiter les images, les algorithmes ici en python, travaillent le plus souvent sur la manipulation des pixels des images. Pour cela, les indices des pixels sont vus comme les coordonnées spatiales de l'image (Figure 1).

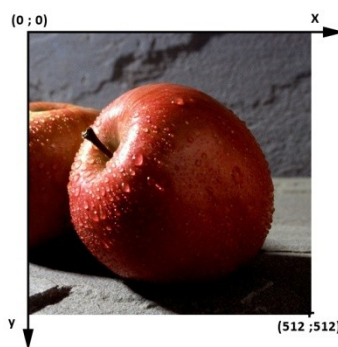


Figure 1. Les coordonnées des pixels.

II. Un pixel¹

Travail à faire

1. Lire un pixel.

- Ouvrir Notepad++.
- Tester le programme suivant :

```
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)
```

¹ www.pixees.fr

- Ecrire le résultat du programme et expliquer-le.

Ce programme lit un pixel et affiche les 3 canaux rouge, vert et bleu du pixel de coordonnées (100 ; 250) de l'image « pomme.jpg ».

Voici une analyse ligne par ligne du programme ci-dessus :

- « from PIL import Image » : une extension (appelé bibliothèque) est nécessaire pour travailler sur les images avec Python. Cette bibliothèque se nomme PIL.
- « img = Image.open("pomme.jpg") » c'est grâce à cette ligne que nous précisons que nous allons travailler avec l'image « pomme.jpg ». Pour travailler avec une autre image, il suffit de remplacer « pomme.jpg » par un autre nom (attention, votre fichier image devra se trouver dans le même dossier que le fichier de votre programme Python).
- « r,v,b=img.getpixel((100,250)) » cette ligne récupère les valeurs du canal rouge (r), du canal vert (v) et du canal bleu (b) du pixel de coordonnées (100 ; 250). Dans la suite du programme, r correspondra à la valeur du canal rouge, v correspondra à la valeur du canal vert et b correspondra à la valeur du canal bleu.
- « print("canal rouge : ",r,"canal vert : ",v,"canal bleu : ",b) » permet d'afficher le résultat.

Travail à faire

2. Lire un autre pixel.

- Modifier le programme précédant pour lire le pixel de coordonnées (150 ; 58).
- Ecrire les lignes du programme modifié.

Travail à faire

3. Modifier un pixel.

- Tester le programme suivant :

```
from PIL import Image
img = Image.open("pomme.jpg")
img.putpixel((256,256),(0,255,0))
img.show()
```
- Expliquer le but du programme.

Un pixel vert a remplacé un pixel rouge au centre de l'image. Voici une analyse des lignes du programme ci-dessus :

- « img.putpixel((250,250),(255,0,0)) » permet de colorier le pixel de coordonnées (250,250) en rouge (255,0,0).
- « img.show() » permet d'afficher l'image modifiée.

Travail à faire

4. Modifier un autre pixel.

- Modifier le programme précédant pour mettre en bleu le pixel de coordonnées (200 ; 116).
- Ecrire les lignes du programme modifier.

III. Les modifications

Pour pouvoir traiter plusieurs pixel, il va falloir parcourir les pixels dans l'image les uns après les autres. Pour cela on utilise la boucle « for ». La boucle « for » permet de faire des itérations sur un élément, comme par exemple une chaîne de caractères ou une liste. Mais il est possible de créer une boucle facilement avec range.

Travail à faire

5. Balayage de l'image.

- Tester le programme suivant :

⚠ Un programme en cours d'exécution peut être arrêté en appuyant simultanément sur la touche Ctrl et sur la touche C.

```
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):
        r, v, b = img.getpixel((x, y))
        print("pixel (" + x + "," + y + ") -> rouge : " + r + ", vert : " + v + ", bleu : " + b)
print("fin")
```

- Expliquer le but du programme.

Les instructions qui sont exécutées dans une boucle « for » doivent être indentées, c'est-à-dire qu'elles doivent être décalées vers la droite. Pour effectuer ce décalage dans « Notepad++ », on utilise la touche de tabulation (« Tab »). La dernière instruction « print("fin") », n'étant pas décalée, elle ne sera pas exécutée dans les boucles mais une fois celles-ci terminées. Les deux boucles « for » vont permettre de parcourir l'ensemble des pixels de l'image, (x ; y) étant les coordonnées des pixel.

Travail à faire

6. Modification de l'image.

- Tester le programme suivant :

```
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):
        r, v, b = img.getpixel((x, y))
        n_r = v
        n_v = b
        n_b = r
        img.putpixel((x, y), (n_r, n_v, n_b))
img.show()
```

- Faire valider par le professeur.

Travail à faire

7. Inversion de canaux.

- Modifier le programme précédent pour inverser les canaux rouge et vert sans changer le bleu.
- Faire valider par le professeur.

Travail à faire

8. Le négatif.

- Modifier le programme pour afficher le « négatif » (une recherche sur le web de la définition sera peut être nécessaire).
- Ecrire les lignes de code.

Travail à faire

9. Le niveau de gris.

- Après avoir fait quelques recherches sur les « images en niveau de gris », écrire un programme qui transforme une « image couleur » en une « image en niveau de gris ».
- ⚠ En Python pour avoir une division entière (le résultat est un entier), il faut utiliser l'opérateur // à la place de l'opérateur / .
- Ecrire les lignes de code.

Travail à faire

10. Une condition.

- Tester le programme suivant :

```
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):
        r,v,b=img.getpixel((x,y))
        if b<200:
            n_b=255-b
        else :
            n_b=b
        img.putpixel((x,y),(r,v,n_b))
img.show()
```
- Expliquer en quelques mots ce que fait ce programme.

Il est même possible de combiner plusieurs conditions :

Travail à faire

11. Deux conditions.

- Modifier la condition par « if b<200 and y>256: ».
- Expliquer en quelques mots ce que fait ce programme et faire valider par le professeur.

IV. Les déplacements

Dans le traitements d'images, il peut être intéressant de déplacer les pixels des images.

Travail à faire

12. Un déplacement.

- Tester le programme suivant :

```
from PIL import Image
```

```
img = Image.open("pomme.jpg")
largeur_image, hauteur_image = img.size
n_img = Image.new(img.mode, img.size)
for y in range(hauteur_image):
    for x in range(largeur_image):
        r, v, b = img.getpixel((x, y))
        n_img.putpixel((x, hauteur_image - y - 1), (r, v, b))
n_img.show()
```

- Expliquer en quelques mots ce que fait ce programme.

Travail à faire

13. Rotation.

- Faire un programme qui réalise une rotation de l'image de 90°.
- Faire valider par le professeur.