

# Projet : Module de manipulation d'image

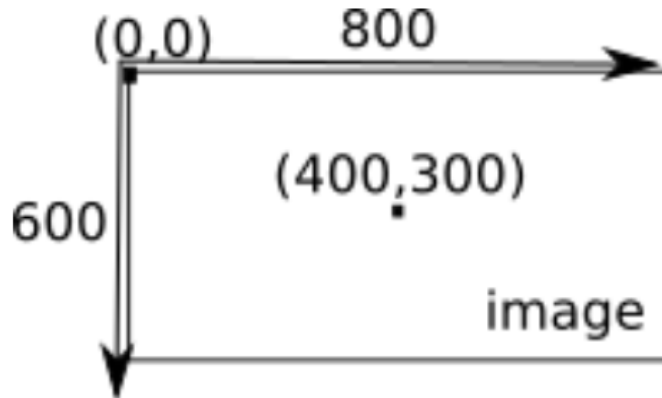
Document basé sur les informations trouvées sur les sites :

- <http://www.tangentex.com/TraitementImages.htm>
- [https://pixees.fr/informatiquelycee/n\\_site/snt\\_photo\\_transimg.html](https://pixees.fr/informatiquelycee/n_site/snt_photo_transimg.html)

## I. Traitement d'image

Pour travailler sur les pixels, il faut travailler la valeur de la composante rouge R, la valeur de la composante vert V et la valeur de la composante bleu B pour un pixel donné.

Avant de commencer à écrire un programme qui nous permettra de travailler sur les pixels d'une image, il est nécessaire de préciser que chaque pixel a des coordonnées (x,y).



Comme vous pouvez le constater sur le schéma ci-dessus, le pixel de coordonnées (0,0) se trouve en haut à gauche de l'image.

Si l'image fait 800 pixels de large et 600 pixels de haut, le pixel ayant pour coordonnées (400,300) sera au milieu de l'image.

Dans un premier temps nous allons utiliser une simple photo de pomme pour faire nos premiers essais, ensuite, vous pourrez travailler avec l'image de votre choix. L'image de la pomme est téléchargeable ici :

[https://ninoo.fr/LC/Term\\_NSI/seqV\\_projet\\_manipulation\\_image/pomme.jpg](https://ninoo.fr/LC/Term_NSI/seqV_projet_manipulation_image/pomme.jpg)

ou (si cela ne marche pas) :

[https://ninoo.fr/LC/Term\\_NSI/seqV\\_projet\\_manipulation\\_image/pomme.png](https://ninoo.fr/LC/Term_NSI/seqV_projet_manipulation_image/pomme.png)

**ATTENTION ! Cette image devra se trouver dans le même dossier que vos programmes Python.**

### A faire vous-même 1.

Saisissez et testez le programme suivant :

```
from PIL import Image
img = Image.open("pomme.jpg")
r,v,b=img.getpixel((235,252))
print("canal rouge : ",r,"canal vert : ",v,"canal bleu : ",b)
```

Ce programme vous donne le canal rouge, le canal vert et le canal bleu du pixel de coordonnées (235, 252) de l'image pomme.jpg

Voici quelques explications :

- `img = Image.open("pomme.jpg")` Ici nous créons un objet image
- `r,v,b=img.getpixel((100,250))` Ici nous récupérons les niveaux de Rouge, de Vert et de Bleu.

### A faire vous-même 2.

- Modifiez le programme du A faire vous-même 1 pour qu'il affiche les valeurs du canal rouge, du canal vert et du canal bleu du pixel de coordonnées (271, 142), notez votre réponse.

.....  
.....

### A faire vous-même 3.

Il est possible de modifier les canaux RVB d'un pixel :

- Recopiez le script suivant afin de modifier la couleur du pixel du centre de vert en rouge.

```
from PIL import Image
img = Image.open("pomme.jpg")
img.putpixel((250,250), (255,0,0))
img.show()
```

### A faire vous-même 4.

- Modifiez le programme du À faire vous-même 3 afin de colorier le pixel de coordonnées (100,250) en bleu.

Modifiez un pixel c'est déjà bien, mais comment faire pour modifier plusieurs pixels ? La réponse est simple, nous allons utiliser des boucles `for`. Nous allons pouvoir balayer toute l'image et ne plus nous contenter de modifier les pixels un par un.

### A faire vous-même 5.

Saisissez et testez le programme suivant

```
from PIL import Image
img = Image.open("pomme.jpg")
for y in range(200, 250):
    for x in range(260,295):
        img.putpixel((x,y), (255,0,0))
img.show()
```

Quelques commentaires sur ce programme :

- Les 2 boucles `for` nous permettent de parcourir un ensemble de pixels de l'image :
  - `for y in range(200, 250):`
  - `for x in range(260,295):`

### A faire vous-même 6.

Compliquons un peu la chose en modifiant tous les pixels de l'image. Nous voulons transformer toute l' image en niveaux de gris.

Il s' agit, pour chaque pixel, de calculer la moyenne des niveaux de rouge, vert et bleu et d' affecter cette valeur à ce pixel.

A noter : en Python pour avoir une division entière (le résultat est un entier), il faut utiliser l'opérateur `//` à la place de l'opérateur `/`

- Complétez et testez le programme suivant :

```
from PIL import Image
img = Image.open("pomme.jpg")
...
...
for y in range(hauteur_image):
    for x in range(largeur_image):
        r,v,b=img.getpixel((x,y))
        gris = ...
        img.putpixel...
img.save("pomme_niveaux_gris.jpg")
img.show()
```

### A faire vous-même 7.

- Après avoir fait quelques recherches sur le "négatif d'une image", modifiez votre programme pour qu' il donne le négatif de l'image image en niveau de gris.

### A faire vous-même 8.

- En jouant avec les coordonnées x,y de chaque pixel, modifier le programme ci-dessous afin qu' il crée une nouvelle image qui est le symétrique suivant un axe vertical de la première.

```
from PIL import Image
img = Image.open("pomme.jpg")
...
...
img2=Image.new('RGB', (largeur_image, hauteur_image))
```

```

for y in range(hauteur_image):
    for x in range(largeur_image):
        r,v,b=img.getpixel((x,y))
        img2.putpixel...
img2.save("pomme_symetrique_v.jpg")
img.show()
img2.show()

```

#### A faire vous-même 9.

- En jouant avec les coordonnées x,y de chaque pixel, modifiez le programme afin qu'il crée une nouvelle image qui est la symétrique suivant un axe horizontal de la première.

#### A faire vous-même 10.

- En jouant avec les coordonnées x,y de chaque pixel, modifiez le programme afin qu'il crée une nouvelle image qui est la rotation de 90° dans le sens des aiguille d'une montre de la première.

#### A faire vous-même 11.

- Complétez le programme afin qu'il demande à l'utilisateur s'il veut faire une rotation de +90°, de +180°, de +270°
- Modifiez le programme afin qu'il crée une nouvelle image qui est la rotation demandée.

## II. Détection des contours

### A. Principes

Tu vas découvrir la détection de contours. Elle permet, par exemple, de repérer des discontinuités dans la profondeur de l'image, de repérer des zones dans une image, de reconnaître des objets

#### A faire vous-même 12.

Tracez le contour que tu visualises sur le carré de valeurs en niveaux de gris. Sélectionne les pixels.



|     |     |     |     |     |
|-----|-----|-----|-----|-----|
| 48  | 49  | 46  | 42  | 44  |
| 110 | 79  | 54  | 47  | 48  |
| 190 | 192 | 190 | 153 | 99  |
| 150 | 166 | 189 | 203 | 183 |
| 131 | 140 | 145 | 161 | 165 |



Un contour se matérialise par une rupture (discontinuité) d'intensité dans l'image suivant une direction donnée.

Prenons un pixel P de coordonnées x,y : P(x,y). Si le niveau de gris des pixels voisins est différent du pixel P(x,y), alors on considérera que ce pixel fait partie d'un contour.

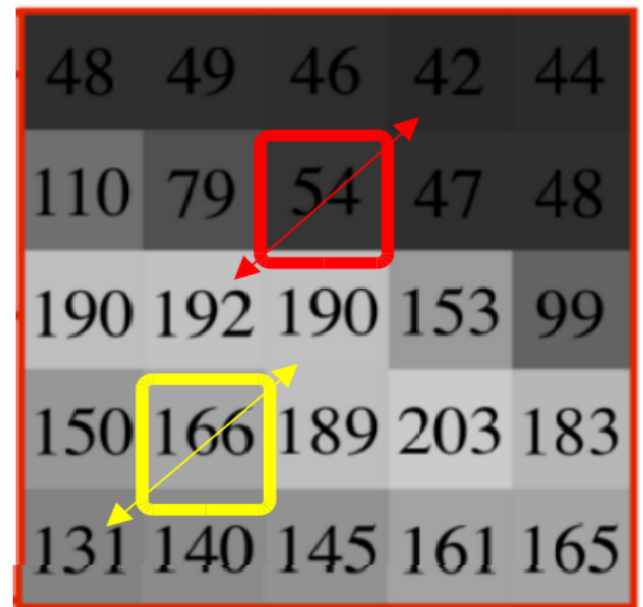
Pour cela il faut donc calculer une différence qui soit la plus significative. Il faudra donc prendre les 2 pixels qui sont diamétralement opposés au pixel central P(x,y). On va donc faire une soustraction entre les deux niveaux de gris des pixels voisins diamétralement opposés.

Quelle est cette différence pour le pixel  $P(x,y)$  encadré en rouge (et de valeur 54). Écris l'opération effectuée ?

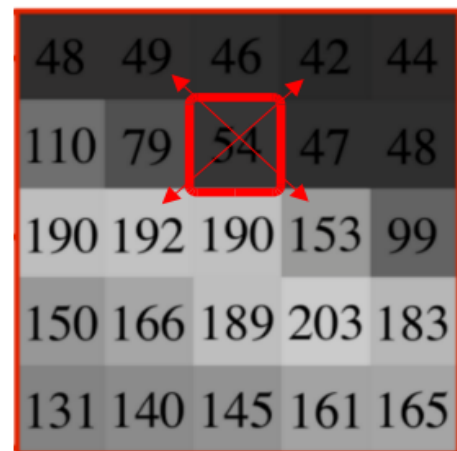
Même question pour le pixel encadré en jaune (de valeur 166).

Parmi ces deux pixels, lequel ou lesquels semble(nt) appartenir à un éventuel contour et pourquoi ?

Pour éviter d'avoir un signe négatif dans la soustraction, il suffira d'élever au carré le résultat qui donnera toujours un résultat positif.



Nous n'avons pris en compte que 2 pixels voisins. Cela privilégie qu'une seule direction. Pour éviter cela on va donc prendre 4 pixels. Refais tous tes calculs de la même façon que précédemment pour les 2 pixels rouge et jaune, mais en prenant compte maintenant 4 pixels. Tu obtiendras 2 résultats (soustraction élevée au carré) que tu additionneras.



## B. Programme python :

A faire vous-même 13.

- Complétez le programme suivant pour faire de la détection de contour comme expliqué ci-dessus :

```
from PIL import Image
img = Image.open("image_niveaux_gris.jpg")
...
valeur_seuil = 40
valeur_seuil = valeur_seuil*valeur_seuil+valeur_seuil*valeur_seuil
img2 = Image.new("RGB", (largeur_image, hauteur_image))
for y in range...
    for x in range...
        ...
        ...
        ...
        norme = ...
        if norme > valeur_seuil:
            img2.putpixel...
        else:
            img2.putpixel...
img2.show()
```

- Testez-le pour plusieurs valeurs de seuil

A faire vous-même 14. (pour les costauds)

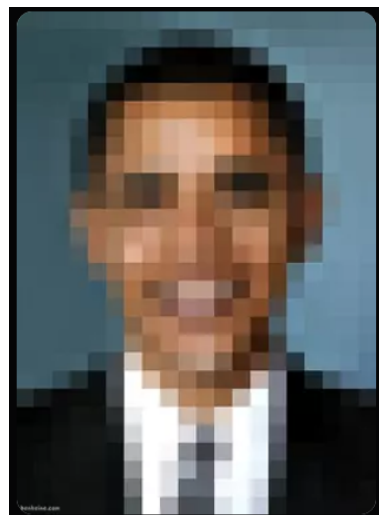
- Écrivez un programme complet qui reprends tout ce que vous avez fait depuis le début
- Le corps de programme est un menu qui demande ce qu'il faut à l'utilisateur et qui crée les objets images

- Le reste est déporté dans des fonctions

### III. PROJET A REMETTRE

A vous appuyant sur le module PIL, proposez plusieurs fonctions permettant de modifier une image.

1. Écrivez une fonction permettant de faire une symétrie suivant l'axe horizontal
2. Écrivez une fonction permettant de faire de la détection de contour
3. Soyez original, écrivez une fonction de votre choix permettant de faire un travail sur les couleurs (éclaircissement, contraste, renforcement de certaines couleurs, ...)
4. Soyez original, écrivez une fonction de votre choix permettant de faire un travail sur les positions des pixels (trucage avec fond vert, image dans image, ...)



5. Écrivez un corps de programme proposant un menu permettant d'exécuter les fonctions :  
*Que voulez vous faire ?*  
 0- Pour quitter  
 1- Pour ...  
 2- Pour ...