

Traitement de données en tables : Manipulation d'images

1. Introduction

Les images sont numérisées selon différents formats (jpeg, bitmap, png...). Une image est un tableau à deux dimensions de pixels dont on peut paramétrer les intensités de couleurs. Plusieurs paramétrages sont possibles en fonction du type de format, du type de couleurs de l'image : en niveau de gris, en mode RGB, ... Les modules Pillow et Numpy permettent de convertir un fichier image en un tableau 2D de pixels manipulables avec python.

Ce TD a pour objectif de vous apprendre à mieux manipuler des tables de données et des tableaux avec python. Le contexte des images permet d'envisager de beaux projets sur ce thème. Pour en savoir plus sur la bibliothèque Pillow, [cliquez ici](#).

2. Images en niveau de gris

Copie dans ton éditeur le code ci-dessus et enregistre-le dans le même répertoire que le fichier Julia.bmp.

```
from PIL import Image
import numpy as np

image = Image.open("Julia.bmp")

#affichons les dimensions de l'image, son format et son mode
nbColonnes,nbLignes = image.size
print("format:", image.format)
print("mode:", image.mode)
print("les dimensions de l'image sont:\n{} pixels en hauteur et {} pixels en largeur".format(nbLignes,
nbColonnes))

julia = np.array(image) #on convertit l'image en un tableau 2D de pixels python
for ligne in julia:
    print(ligne)
```

L'exécution de ce code nous renseigne sur le format de l'image Julia.bmp, son mode et ses dimensions.

De plus en observant l'affichage du tableau julia, on constate que la couleur de chaque pixel est stockée sous la forme d'un tableau [r, g, b] où r, g et b reçoivent la même intensité [niveauGris, niveauGris, niveauGris].

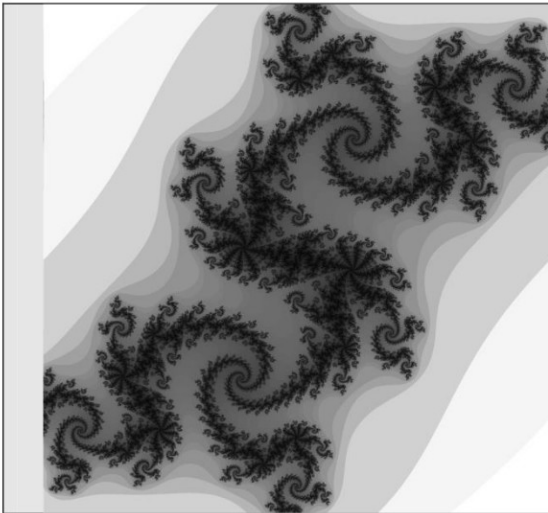
Ainsi pour le triplet [0,0,0] le pixel est noir, pour [255,255,255] il est blanc. Chacune des intensités Red, Green et Blue sont des nombres compris entre 0 et 255 ici.

Exercice 1. Complète le code ci-après pour créer un nouveau fichier bitmap contenant le négatif de l'image Julia.bmp . Utilise le complément à 255 pour redéfinir les niveaux de gris.

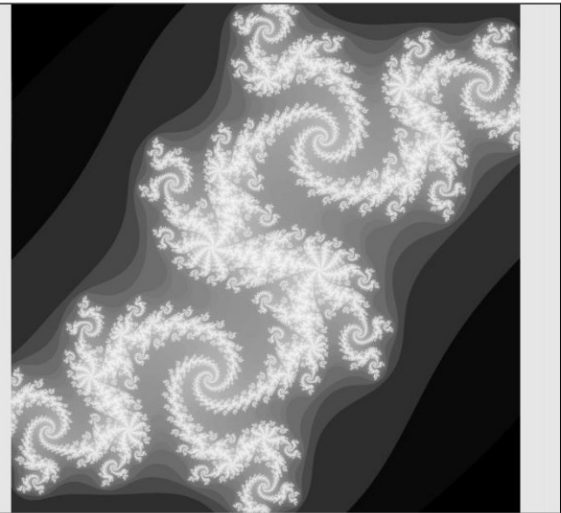
```
from PIL import Image
import numpy as np
image = Image.open("Julia.bmp")
nbColonnes,nbLignes = image.size
julia = np.array(image)
```

```
#on convertit au format bitmap et on enregistre le nouveau fichier
newImage=Image.fromarray(julia)
newImage.save('négatif_Julia.bmp')
```

Julian.bmp



négatif_Julia.bmp



3. Images en couleurs et Mode RGB

Exercice 2. Écris un programme qui permette d'ouvrir le fichier Maryline.bmp et de le modifier afin d'obtenir son négatif couleur.

Pour cela il suffit de redéfinir chacun des pixels en utilisant le complément à 255 pour qu'il affiche sa 'couleur négatif'.

Image d'origine

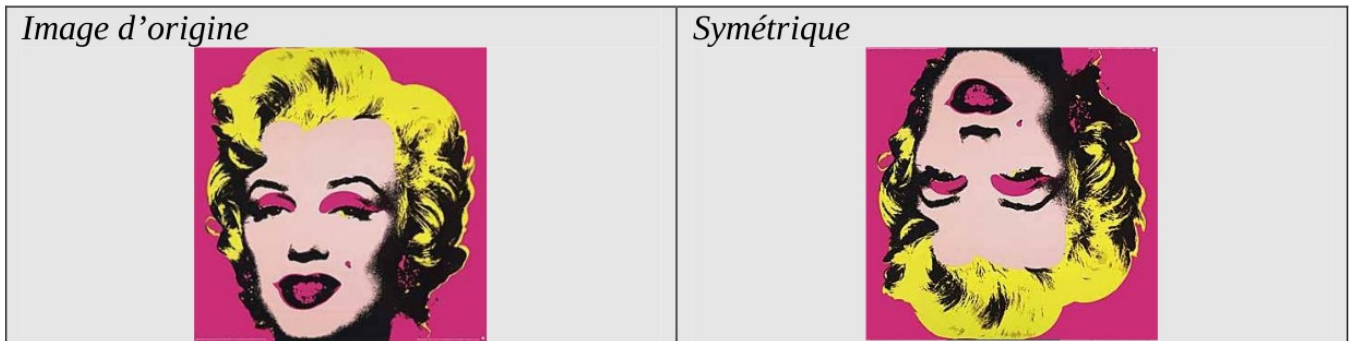


Négatif



Exercice 3. Écris un programme permettant de générer l'image symétrique de l'image de Marylin.

Tu dois obtenir :



Exercice 4. Écris un programme permettant de générer l'image miroir de l'image de Marylin. Il s'agit ici d'obtenir l'image symétrique par rapport à la verticale passant par le centre de l'image

Exercice 5. De la couleur au niveau de gris

Pour générer une image en niveau de gris à partir d'une image en couleurs, il suffit de remplacer chaque triplet (r,g,b) par le niveau de gris obtenu en calculant la moyenne $\frac{r+g+b}{3}$. On prendra la partie entière de cette valeur pour définir le niveau de gris (voir exercice 1) de chacun des pixels.

Écris un programme permettant d'obtenir l'image en niveau de gris de Marylin.

Résultat attendu :

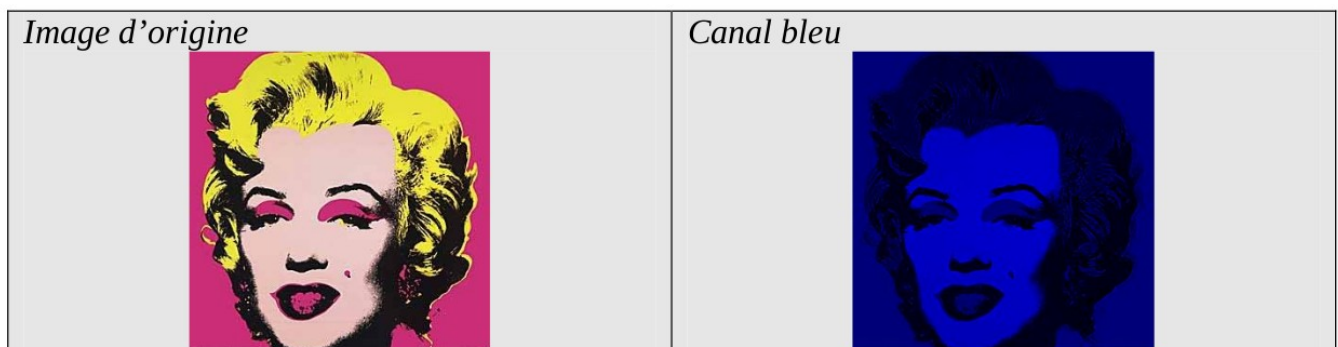


Exercice 6. Les canaux RGB

Une image en couleurs est en réalité composée de trois images, une première n'utilisant que le canal rouge, la deuxième utilisant le canal vert et la dernière le canal bleu.

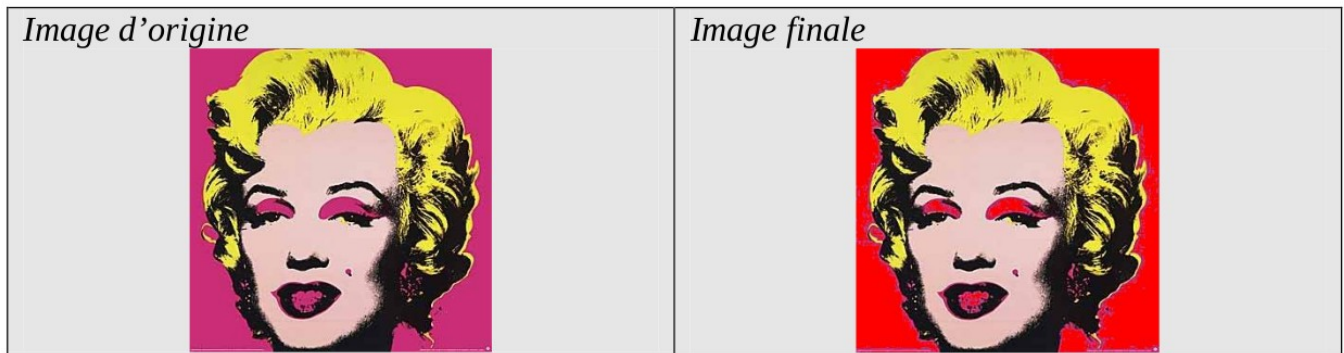
Écris un programme permettant d'obtenir les 3 images dans chacun des 3 canaux R, G et B.

Pour le canal bleu tu dois obtenir :



Exercice 7. a) Modifier une palette de couleurs 1

Écris un programme permettant de passer de l'image de gauche à celle de droite :



b) Modifier une palette de couleurs 2



Écris un programme permettant d'obtenir le taureau ci-dessus à partir du fichier taureau-de-duhem.png. Attention, au format png, l'état d'un pixel est donné sous la forme d'un quadruplets [r, g, b, intensité].

4. Modifier les dimensions d'une image

Exercice 8. Diminuer la taille d'une image

Écris un programme qui génère une miniature de l'image poivron.bpm dont la longueur des cotés mesurent la moitié de celles de l'image initiale. Précision utile : les dimensions de l'image étant divisées par 2, la surface de l'image est divisée par 4....par conséquent, l'image réduite aura 4 fois moins de pixels que celle d'origine



Exercice 9.

Écris un programme qui permette d'obtenir l'image de droite à partir de celle de gauche.
Précisions : l'image produite doit avoir les mêmes dimensions que l'image de départ à 2 pixels près en largeur et en longueur.

Image d'origine de côté 512

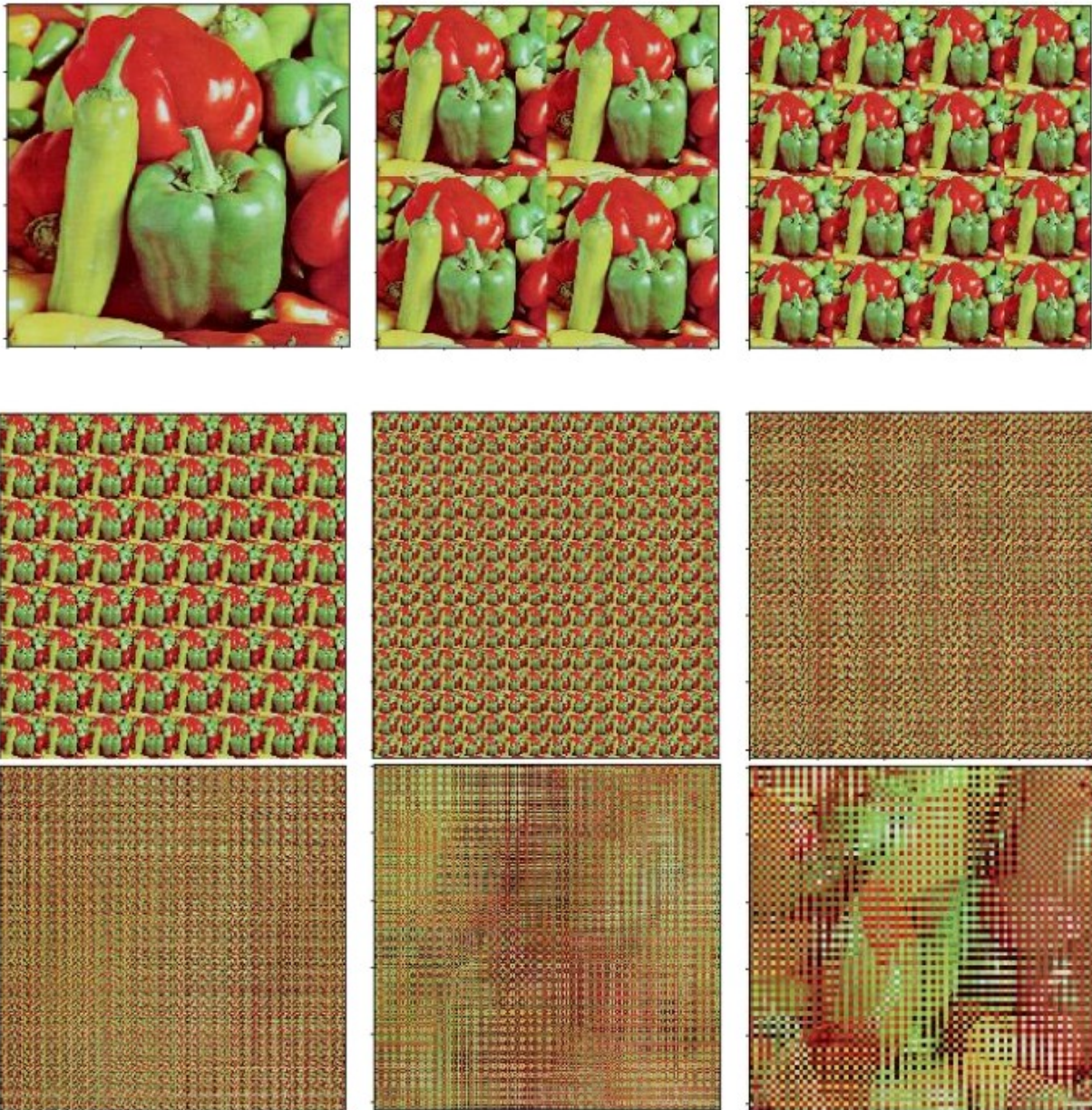


Image finale de côté 512 constituée de quatre miniatures identiques



5. Mini-projet : Le photomaton

En vous aidant des exercices précédents, écrire un programme qui affiche successivement les différentes images d'un photomaton. Voici la séquence des images que doit pouvoir générer votre programme si on lui fournit l'image poivron.bpm :



Votre programme pourrait gérer des images qui ne sont pas nécessairement carrées et proposer des qualités et dimensions d'images en sorties qui soient paramétrables par l'utilisateur.

Pour l'affichage vous pouvez utiliser la bibliothèque matplotlib :

```
#importer la bibliothèque
import matplotlib.pyplot as plt
# et les méthodes à placer dans le code pour afficher l'image obtenue après transformation
plt.imshow(tableau2D)
plt.show()
```