La retouche d'images par programmation python

La statue … d'après le travail de M. CANTUS

1. Introduction

Nous allons travailler avec deux fichiers : l'image **statue.jpg** et le fichier **modifier.py** qui doivent être enregistrés dans le même dossier.

Important : la définition de l'image est de **400 × 500** pixels. Ainsi les numéros des colonnes vont de 0 à 399 et les numéros des lignes de 0 à 499.

Dans une première partie nous travaillerons sur les changements de couleurs de l'image, dans une deuxième partie sur les changements géométriques de l'image.

2. Changements de couleur

2.1. La composante rouge

Dans un premier temps, nous allons créer une nouvelle image qui sera la composante rouge de l'image originale, c'est-à-dire que dans les triplets de couleurs, on garde la valeur du rouge, on passe celles du vert et du bleu à 0. Par exemple, le tuple (142, 184, 200) deviendra (142, 0, 0).

1. Remettre les instructions suivantes dans l'ordre pour obtenir un algorithme permettant de créer l'image avec la modification souhaitée.

A - Créer la couleur modifiée de triplet (R,0,0)
B - Lire le triplet (R,V,B) du pixel de l'image dont les coordonnées sont (c,l)
C - Fin Pour
D - Pour chaque pixel de l'image
E - Colorer le pixel de la nouvelle image dont les coordonnées sont (c,l) avec la couleur modifiée

2. Lire les lignes 10 à 20 du code suivant.

```
1  # Importation des librairies utilisées
   from PIL import Image
   """ Ne pas modifier : déclaration des images."""
   nom="statue.jpg"
   image=Image.open(nom)
   largeur,hauteur=image.size
   nouvelleImage=Image.new("RGB",(largeur,hauteur))

   """ Création de l'image modifiée """
10 for l in range (0,500):
       for c in range (0,400):
           # Lecture du pixel de base
           couleur=image.getpixel((c,l))
           # modification
           couleur_mod=(couleur[0],0,0)
           # Recopie
           nouvelleImage.putpixel((c,l),couleur_mod)


20 """ Ne pas modifier : sauvegarde puis affichage de l'image."""
   nouvelleImage.save("Image rouge.png")
   nouvelleImage.show()
```

a. Dans quel ordre les pixels sont-ils parcourus ?
b. Que font les lignes 13, 15 et 17 par rapport à l'algorithme de la question 1 ?

3. Ouvrir le fichier **modifier.py** et sauvegarder le script sous le nom **image_rouge.py** puis lancer le script en cliquant sur la flèche verte.

Retour sur la ligne 15. Nos triplets (R,V,B) sont donc codés à l'aide de tuples à 3 éléments. Pour l'ordinateur, les éléments sont numérotés avec l'ordre 0, 1 et 2.
Prenons en exemple le tuple : `couleur=(142, 184, 200)`.

- 142 est l'élément numéro 0 du tuple `couleur`. On le note `couleur[0]`.
- 184 est l'élément numéro 1 du tuple `couleur`. On le note `couleur[1]`.
- 200 est l'élément numéro 2 du tuple `couleur`. On le note `couleur[2]`.

Cette notation reste valable quelles que soient les valeurs du triplet.

2.2. L'image en cyan

Penser à :

- ouvrir le fichier **modifier.py** et sauvegarder le script sous un nom adapté ;
- modifier le nom de l'image créée en ligne 21 pour la retrouver facilement.

Coder et créer l'image en composante cyan : le triplet (142, 184, 200) deviendra (0, 184, 200).

2.3. L'image en inversion de couleurs

Penser à :

- ouvrir le fichier **modifier.py** et sauvegarder le script sous un nom adapté ;
- modifier le nom de l'image créée en ligne 21 pour la retrouver facilement.

Coder et créer l'image en échangeant les niveaux de rouge et de bleu : par exemple, le triplet (142, 184, 200) deviendra (200, 184, 142).

2.4. L'image en négatif

Penser à :

- ouvrir le fichier **modifier.py** et sauvegarder le script sous un nom adapté ;
- modifier le nom de l'image créée en ligne 21 pour la retrouver facilement.

Coder et créer l'image en remplaçant chacune des 3 composantes par son complément à 255.
Ainsi, le triplet (100, 200, 50) deviendra (155, 55, 205).

2.5. L'image en niveaux de gris

Penser **à chaque fois** :

- ouvrir le fichier **modifier.py** et sauvegarder le script sous un nom adapté ;
- modifier le nom de l'image créée en ligne 21 pour la retrouver facilement.

Coder et créer une image en gris revient à mettre les 3 composantes R, V et B à la même valeur. Nous allons créer 3 images en niveaux de gris à partir de la même image afin de comparer les différentes techniques.

2.5.1. Gris sur la composante rouge

Choisir de mettre les 3 valeurs au niveau de celle de la composante rouge. Par exemple, le tuple (142, 184, 200) deviendra (142, 142, 142).

2.5.2. Gris sur la valeur maximale

Choisir le niveau maximal des trois valeurs. Par exemple, le tuple (142, 184, 200) deviendra (200, 200, 200).
→ L'instruction `max(a,b,c)` renvoie la plus grande valeur parmi les nombres `a`, `b`, et `c`.

2.5.3. Gris sur la valeur moyenne

Choisir le niveau moyen des trois valeurs. Par exemple, le tuple (142, 184, 200) deviendra (175, 175, 175) car 175 est la valeur arrondie à l'entier de la moyenne des nombres 142, 184 et 200.
→ L'instruction `int(a/3)` renvoie l'arrondi par défaut de la division du nombre a par 3.

2.5.4. Gris sur la valeur naturelle

Un triplet (R, V, B) est remplacé par du gris calculé avec la formule :

$$\mathbf{0{,}2126 \times R + 0{,}7152 \times V + 0{,}0722 \times B}$$

Penser à arrondir si nécessaire !

Ainsi, le triplet (142, 184, 200) donne le calcul :

$0{,}2126\times142+0{,}7152\times184+0{,}0722\times200 = 176{,}226$ qu'on arrondit par défaut à 126

et deviendra (126, 126, 126).

Comparer les 4 images obtenues.

3. Effets géométriques

3.1. Symétrie par rapport à un axe vertical

Penser à :

- ouvrir le fichier **modifier.py** et sauvegarder le script sous un nom adapté ;
- modifier le nom de l'image créée en ligne 21 pour la retrouver facilement.

L'ordre des pixels d'une ligne est inversé.

Exemple avec la ligne numéro 0 : le pixel de coordonnées (0, 0) se retrouve en (399,0), celui de coordonnées (1, 0) se retrouve en (398, 0), …, celui de coordonnées (399, 0) se retrouve en (1, 0).

3.2. Symétrie par rapport à un axe horizontal

Penser à :

- ouvrir le fichier **modifier.py** et sauvegarder le script sous un nom adapté ;
- modifier le nom de l'image créée en ligne 21 pour la retrouver facilement.

L'ordre des pixels d'une colonne est inversé.

Exemple avec la colonne numéro 0 : le pixel de coordonnées (0, 0) se retrouve en (0, 499), celui de coordonnées (1, 0) se retrouve en (0, 498), …, celui de coordonnées (0, 499) se retrouve en (0, 0).

3.3. Rayures horizontales

Penser à :

- ouvrir le fichier **modifier.py** et sauvegarder le script sous un nom adapté ;
- modifier le nom de l'image créée en ligne 21 pour la retrouver facilement.

Chaque ligne de numéro pair est conservée, les lignes de numéro impair sont noires.