Traitement d'images

Ce TD propose de s'initier à travers quelques exercices au travail sur les images avec python.

1 Objets et méthodes pour les images en python

Une image numérique peut prendre le format d'un tableau de pixels. On travaillera en python avec ce type d'images.

1.1 Importation d'une image

Pour travailler avec des images que l'on a sur le disque dur de l'ordinateur, il faudra exécuter la commande :

```
from PIL import Image, ImageFile
```

Vous pouvez dès à présent ouvrir un éditeur python et écrire ceci au début de votre fichier.
Après avoir téléchargé le fichier *perroquet.png* sur le site de la PCSI, assurez vous de l'enregistrer dans le répertoire où est également le fichier que vous êtes en train d'éditer en python. Vous pouvez alors ajouter la commande suivante dans votre fichier pour récupérer votre image dans une variable avec laquelle vous travaillerez ensuite :

```
img = Image.open("perroquet.png")
```

Ici, on donne le nom *img* à l'image mais vous pouvez bien sûr utiliser d'autres noms de variables.

1.2 Méthodes pour lire ou modifier un pixel

Si l'image est de largeur l pixels et de hauteur h pixels et stockée dans la variable *img* par exemple, on pourra accéder à la couleur du pixel de la colonne d'indice $k \in [\![0, l-1]\!]$ et la ligne d'indice $j \in [\![0, h-1]\!]$ de l'image à l'aide de la méthode :

```
img.getpixel((k,j))
```

Cette méthode renvoie un triplet (r, v, b) de nombres. Le plus simple est donc de l'utiliser sous la forme :

```
r,v,b=img.getpixel(k,j))
```

Ici, l'image du perroquet est un carré de 512 pixels de côté. Les entiers k et j pourront donc prendre les valeurs 0 à 511.
Pour changer la couleur d'un pixel de l'image, on pourra utiliser la méthode :

```
img.putpixel((x,y),(r,v,b))
```

où r, v et b doivent être des nombres compris entre 0 et 255.

1.3 Méthodes générales

Pour créer une image blanche, on pourra utiliser :

```
img = Image.new('RGB', (l,h))
```

où l et h doivent être des entiers qui seront respectivement la largeur et la hauteur de l'image en pixels. Bien sûr, on peut utiliser un autre nom que *img* pour l'image blanche ainsi créée.
Pour connaître les dimensions d'une image, on pourra utiliser :

```
l , h=img . s i z e
```

afin d'obtenir la largeur dans la variable l et la hauteur comme valeur de h.
Enfin, pour afficher l'image *img*, on utilisera la commande :

```
img . show ( )
```

2 Exercices

2.1 Bandes colorées

1. Remplacer les 50 premières lignes de pixels de l'image *perroquet* par des pixels tous rouges afin d'obtenir une bande horizontale rouge en haut de l'image.
2. Remplacer les 50 premières colonnes de pixels de l'image *perroquet* par des pixels verts.

2.2 Miroirs

1. Réaliser une symétrie de l'image par rapport à l'axe vertical situé au milieu, c'est à dire ici comme les pixels ont leurs abscisses numérotées de 0 à 511 que les couleurs initiales du pixel de coordonnées (x, y) seront les couleurs finales du pixel d'abscisse $(511 - x, y)$ (et inversement).
2. Réaliser alors une symétrie par rapport à l'axe horizontal situé au milieu de l'image.

2.3 Décompositions de l'image

1. Extraire de l'image exclusivement les teintes de rouge, c'est à dire que tout pixel de couleur (r, v, b) se voit remplacé par $(r, 0, 0)$
2. Extraire de même les images rouges et vertes issues du perroquet.
3. Réaliser le « négatif »de l'image : un pixel de couleur (r, v, b) se voit colorié en $(255 - r, 255 - v, 255 - b)$.

2.4 Images aléatoires

Pour pouvoir faire cet exercice, on écrira préalablement la ligne :

```
from random import randint
```

afin de disposer de la fonction *randint(a,b)* qui renvoie un entier pseudo-aléatoire de l'intervalle $[\![a, b]\!]$.

1. Ecrire une fonction $aleatoire(l, h)$ qui crée une image de largeur l et de hauteur h dont tous les pixels ont leurs trois nuances de rouge, vert et bleu obtenues à l'aide de la fonction *randint(0,255)*.
2. Ajouter une sorte de bruit sur une image : écrire une fonction *bruit(img,p)* où *img* est une image et p un nombre entre 0 et 1. On génèrera une image aléatoire de la même taille que *img* et l'on renverra enfin une image dont la couleur du pixel de coodonnées (x, y) est obtenue par une moyenne entre le pixel de *img*, disons $(r1, v1, b1)$ et l'image aléatoire créée, disons $(r2, v2, b2)$, selon la formule :

   ```
   ((1 - p)* r1+p*r2 , (1 - p)* v1+p*v2 , (1 - p)* b1+p*b2 )
   ```