Traitement de l'image

I) La bibliothèque PIL

PIL pour (python imaging library) est une bibliothèque python qui permet de faire du traitement d'image.
http:/www.pythonware.com/products/pil/

Instructions de base.

```
from PIL import Image
im = Image.open("fichier.ext")
```

Ceci importe la partie Image de la librairie PIL, et charge un fichier image, et crée un objet (appelé im, ici) qui contient les informations de l'image. Il faut que le fichier python soit enregistré dans le même répertoire que l'image.

L'objet im a alors plusieurs attributs :
im.size est un couple (tuple en python) qui donne la largeur puis la hauteur de l'image en pixel.
im.mode donne le mode colorimétrique de l'image. Le mode est donné sous forme d'une chaîne de caractères. Les modes courants sont :

- 'L' (pour luminance) : niveau de gris. Chaque pixel est codé par une valeur numérique sur un octet, de 0 (noir) à 255 (blanc).
- 'RGB' (Red, Green, Blue) : image en couleur, chaque pixel est codé par un triplet de valeurs numériques (sur un octet chacune).
- 'CYMK' (cyan, jaune, magenta, noir) : pour des images destinées à l'impression.
- '1' : noir ou blanc, chaque pixel est codé par une valeur binaire 0 (noir) ou 1 (blanc).

im.format renvoie le format de l'image qui a été chargée (bmp, jpg…). C'est vide si l'image n'a pas été chargée depuis un fichier. Les principaux formats d'image sont traités.

On peut créer une image *ex nihilo*, avec l'instruction Image.new(mode, size, [couleur]), en précisant le mode (via une chaîne de caractères), la taille (via un tuple) et, si on le souhaite la couleur des pixels (par défaut, les pixels sont tous noirs.

On peut afficher une image par la méthode .show(), qui affichera l'image dans le logiciel configuré par défaut.
On peut enregistrer une image par la méthode .save("nom du fichier.ext"), fichier qui sera enregistré dans le répertoire du fichier python.

Pour avoir accès à la liste complète des valeurs des pixels d'une image, on applique la méthode .getdata() (qui, techniquement, ne créée pas une liste, mais un truc qui ressemble à une liste, que l'on peut ensuite convertir en liste).
Pour réaffecter de nouvelles valeurs à tous les pixels d'une image, on peut appliquer la méthode .putdata(liste).

Pour accéder à la valeur d'un pixel de coordonnées (x ; y), on utilise la méthode .getpixel((x,y)) (attention au double parenthésage : l'argument pris par la méthode est un couple donnant les coordonnées du pixel).
Remarque : L'abscisse d'un pixel correspond bien à sa position horizontale sur l'image, le 0 étant à gauche de l'image et les abscisses positives vers la droite, l'ordonnée d'un pixel est sa position verticale, mais attention, le 0 est en haut de l'image et les abscisses positives vers le bas.

Pour modifier la valeur d'un pixel, on utilise .putpixel((x,y),valeur), où valeur peut être un nombre (si le mode de l'image est 1 ou L) ou un tuple (pour les images RGB et CYMK).

II) Traitement d'une image

1) Créer un négatif

Soit pour une image en niveaux de gris, ou bien pour une image en couleur : la valeur de chaque pixel (chaque composante de la valeur d'un pixel) est remplacée par son complément à 255.

2) Créer l'image en niveaux de gris, à partir d'une image couleur

La méthode simple pourrait être de mettre en niveau de gris la moyenne (arrondie à l'entier) des trois composantes de la valeur du pixel en couleur, cependant, l'usage n'est pas celui-ci, la Society of Motion Picture and Television Engineers recommance d'appliquer la formule suivante :
Luminance = 0,2126 × Rouge + 0,7152 × Vert + 0,0722 × Bleu (la somme des coefficients est 1).
La formule est calculée à partir de la courbe d'efficacité lumineuse spectrale de l'observateur de référence de la Commission Internationale de l'Eclairage et des couleurs primaires normalisées sur un type d'affichage vidéo.
(voir https://fr.wikipedia.org/wiki/Niveau_de_gris).

3) Créer une image en sépia, à partir d'une image couleur

C'est un peu le même principe, mais cette fois, l'image créée reste en couleur, on commence par calculer la quantité de vert :
vert' = 0,299 × rouge + 0,587 × vert + 0,114 × bleu.
Puis, si cette quantité de vert est strictement inférieure à 63, alors le rouge est rouge' = vert' × 1,1 et le bleu est bleu' = vert' × 0,9.
Si la quantité de vert est entre 64 et 192, alors le rouge est rouge' = vert' × 1,15 et le bleu est
bleu' = v × 0,85.
Si la quantité de vert est supérieur à 193, alors on applique rouge' = vert' × 1,08 (plafonné à 255) et
bleu' = vert' × 0,93.
(je ne retrouve plus les sources pour cette conversion).

4) Effectuer un seuillage :

On créée une image en noir et blanc, à partir d'une image en niveaux de gris. (L'image en N&B peut rester en mode L, ou bien être en mode 1). On fixe un seuil (ou on demande à l'utilisateur d'entrer un seuil), entre 0 et 255 : tous les pixels dont la luminance est supérieure au seuil seront affichés en blanc, tous les autres en noir.

5) Créer une image plus constrastée.

Pour cela, on va choisir un seuil (30, par exemple) et décider que toutes les valeurs de pixels (soit en niveaus de gris, soit en couleur) inférieures au seuil, ou supérieures à 255 moins le seuil sont mises à 0 ou 255 respetivement, les valeurs intermédiaires sont rééchelonnées entre 0 et 255 de façon affine.

6) Créer une image en ne gardant que la composante rouge de chaque pixel

Cela reproduit l'effet d'une vision à travers un filtre rouge parfait.
Les composantes verte et bleue sont simplement mises à 0.

7) Détection de contours.

L'idée est de comparer la valeur d'un pixel à celles des pixels voisins : si les valeurs sont proches, on va afficher du blanc, si elles sont éloignées, on affiche du noir. Le grand classique est de multiplier par 8 la valeur du piexel central, et de soustraire la somme des valeurs des 8 pixels qui l'entourent. Si la valeur obtenue (en valeur absolue) est proche de 0 (il y a donc un seuil à définir), on considère que les pixels sont de couleurs proches, sinon, on condidère qu'ils sont de couleur éloignées et donc on détecte un contour.
On peut évidemment avoir des zones de détection plus étendues que les 8 pixels qui entourent un pixel central, en choisissant des coefficients adaptés.

8) Effectuer une rotation d'angle droit (sens direct ou indirect), d'angle plat ou une symétrie axiale

Le titre est assez explicite, il faut réfléchir à l'orientation souhaitée et réaffecter les pixels.