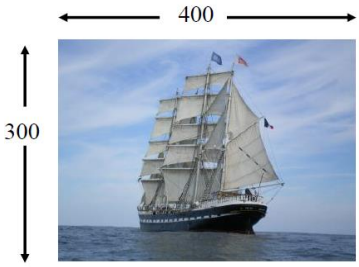
Traitement d'image à l'aide de Python

Présentation d'une image numérique

Cette appellation désigne toute image (dessin, schéma, photographie, …) acquise, créée et stockée sous forme numérique. On ne traitera par la suite que les images matricielles (ou image bitmap) en 2 dimensions (hauteur et largeur).
Chaque point est appelé pixel et est codé sur 1 bit (image noir et blanc) ou plusieurs (image couleur ou en niveau de gris). Quelques caractéristiques :



- La définition : indique le nombre de point en largeur (axe vertical) * le nombre de point en hauteur (axe horizontal). Exemple 400 * 300.
- La résolution : indique le nombre de point par unité de longueur (exemple 400 ppp ou pixel par pouce, soit le nombre de point tous les 2,54 cm). Ce paramètre est choisi lors de la conversion analogique numérique (acquisition) de l'image (par l'appareil photo, le scanner, …).

A l'aide de ces deux paramètres, on définit la taille de l'image.

- La représentation des couleurs :
  - ✓ Par 2 couleurs, le noir et blanc, 1 seul bit suffit à coder l'information
  - ✓ En niveau de gris on utilise alors souvent un octet pour coder les 256 niveaux de gris possibles (0 pour le noir, et 255 pour le blanc). On code ainsi l'intensité lumineuse.
  - ✓ En couleurs (dites vraies) : le plus utilisé est une combinaison des trois couleurs Rouge, Vert et Bleu (synthèse additive). Chaque couleur est codée sur un octet et on obtient ainsi 16,7 millions de couleurs différentes (0,0 et 0 codent le noir alors que 255,255 et 255 codent le blanc).

Création d'une image au format .pbm

But : Créer un fichier image avec un motif particulier. Ce fichierdoit pouvoir être lu soit par un éditeur de texte (notepad++) soit un logiciel de dessin (gimp).

Il est impératif lors de la mémorisation d'une image de respecter un format de fichier. Les plus courants sont le .jpeg, le tiff, le gif, le bmp, le raw, …. Ils se différentient par :

- Le codage de chaque pixel,
- Le standard ouvert ou propriétaire,
- Le mode compressé ou non, destructif ou pas,
- L'ajout d'informations supplémentaires (métadonnées), …

L'un des formats les plus simples à mettre en œuvre est le .pbm (Portable Bitmap File format). Il propose des fonctionnalités élémentaires. Il a été défini par Jef Poskanzer dans les années 1980 comme un format d'images bitmaps monochromes pouvant être transmises via un message électronique en texte ASCII.

Voici un programme :

```
largeur=70
hauteur=70

# Ouverture d'un fichier en mode ecriture
fichier =open( "image.pbm" , "w" )

# ecriture de l'entete du fichier
fichier.write("P1")
# caractere saut de ligne \n
fichier.write("\n")
# dimensions de l'image codée dans le fichier
fichier.write(str(largeur)+" "+str(hauteur))

# Ecriture de l'image pixel par pixel
for h in range(hauteur):
    fichier.write("\n")
    for l in range(largeur):
        if h%2==0:
            fichier.write("0")
        else :
            fichier.write("1")
fichier.close()
```

- ✓ Exécuter ce programme et observer le fichier obtenu (même répertoire que le fichier .py) à l'aide de l'éditeur et du logiciel de dessin.
- ✓ Comment est codé un pixel blanc, et un pixel noir ?
- ✓ Expliquer le rôle des informations présentes dans l'entête. Comment sont elles codées pour pouvoir être lues avec un éditeur de texte ? (http://fr.wikipedia.org/wiki/Portable_pixmap)
- ✓ Modifier ce programme pour obtenir une image de largeur et de hauteur 40 pixels, avec que des traits verticaux.

Il est possible de tracer des formes particulières.

- ✓ Exécuter et observer le programme ci-contre.
- ✓ Quelle est la figure représentée ?
- ✓ A quel endroit du programme est choisie la forme ?
- ✓ Modifier le programme de façon à obtenir une droite reliant le pixel en bas à gauche à celui en haut à droite ?
- ✓ Proposer un programme réalisant un x le plus grand possible ?

```

def fonct(y,x):
    return (y-x)

largeur=70
hauteur=70
f=open( "image2.pbm" , "w" )
f.write("P1")
f.write("\n")
f.write(str(largeur)+" "+str(hauteur))

for h in range(hauteur):
    f.write("\n")
    for l in range(largeur):
        if fonct(h,l)==0 :
            f.write("1")
        else :
            f.write("0")
f.close()
```

Pour aller plus loin :
Tracer un cercle centré dans l'image de rayon 18 pixels et d'épaisseur 5 pixels.

Traitement d'une image à l'aide d'une bibliothèque

La bibliothèque PIL

Lorsque le format de l'image se complique (.bmp, .jpeg, …), le travail nécessaire à la création ou à la modification du fichier image devient long et compliqué. Il est alors judicieux d'utiliser les bibliothèques compatibles avec Python.

But : Modifier un fichier image en utilisant le module PIL de Python.

On utilisera le module PIL (Python Imaging Library). Il supporte plusieurs formats de fichier, parmi lesquels .png, .jpeg, .tiff et .bmp.

Voici ci-contre quelques unes des nombreuses fonctions disponibles.

Pour en savoir plus : (http://pillow.readthedocs.org/en/3.0.x/handbook/tutorial.html).

```
# Pour utiliser la bibliothèque
from PIL import Image

# Ouverture d'un fichier Image
im = Image.open("londres.jpg")
# Permet de visualiser l'image
im.show()

# Pour connaitre les dimensions de l'image
largeur , hauteur = im.size
print (largeur, hauteur)

# Renvoie un tableau 2D contenant les pixels
pixels = im.load()

# Pour lire le pixel connu par ses coordonnées (l,h)
rouge, vert, bleu = pixels[l,h]
# Permet de modifier la couleur d'un pixel
# colorie le pixel en bas à gauche en rouge
pixels[0, hauteur - 1] = (255, 0, 0)

# Sauve les modifications dans un nouveau fichier
im.save( "londres_mod.jpg")

# Pour créer une nouvelle image
# 'RGB' pour une image couleur,
# 'L' pour une image monochrome
# largeur et hauteur pour les dimensions
image=Image.new('RGB',(largeur,hauteur))
```

✓ Ecrire un programme permettant
- d'ouvrir le fichier « londres.jpg »,
- de le visualiser.
- et d'afficher ses dimensions.

✓ Exécuter le programme ci-contre. Analyser le et indiquer quel rôle réalise la fonction ligne().

✓ Renseigner alors l'entête de la fonction.

✓ Modifier cette fonction de façon à tracer une ligne verticale.

```
# -*- coding: utf-8 -*-

def ligne(numero,image) :
    """
        """
    largeur , hauteur = image.size
    image_mod=Image.new('RGB',(largeur,hauteur))
    pixels = image.load()
    pixels2 = image_mod.load()
    for h in range (0,hauteur) :
        if h != numero :
            for l in range (0,largeur):
                pixels2[l,h] = pixels[l,h]
        else :
            for l in range (0,largeur):
                pixels2[l,h] = (0,0,0)

    return(image_mod)


from PIL import Image

im = Image.open("londres.jpg")
im.show()

image_mod=ligne(10,im)
image_mod.show()
```

Les premiers Traitements

- ✓ Ecrire une fonction permettant de réaliser un effet miroir. Cette fonction recevra l'image à modifier et renverra l'image modifiée.


- ✓ Ecrire une fonction convertissant une image couleur en une image noir et blanc. On obtient ainsi le signal de luminance. Chaque pixel reçoit une moyenne coefficientée des trois couleurs. On utilise la formule $G_r = 0.299*R+ 0.587*G + 0.114*B$ (ou G_r, R, G et B sont respectivement les niveaux de gris, rouge, vert et bleu)

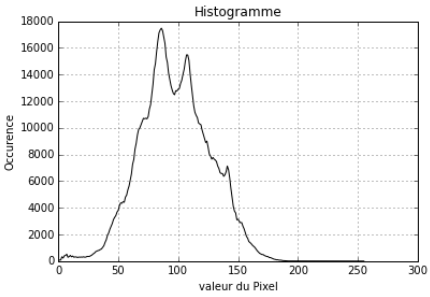
https://fr.wikipedia.org/wiki/Luminance#Luma


- ✓ Ecrire une fonction convertissant une image couleur en son négatif. Rechercher sur internet les opérations à réaliser. Cette fonction recevra l'image à modifier et renverra l'image modifiée.


- ✓ ***Pour aller plus loin : Ecrire une fonction permettant de réaliser une rotation en degré (compris entre 0 et 360°). Cette fonction recevra l'image à modifier et l'angle choisi et renverra l'image modifiée.***

Le filtrage

Un outil très employé pour pouvoir appliquer un filtre sur une image est de réaliser son histogramme. A chaque valeur de couleur, on associe le nombre de pixels ayant cette valeur.
On réalise ainsi 3 histogrammes pour une image couleur ou un seul pour une image noir et blanc.



- ✓ Ecrire une fonction réalisant l'histogramme de l'image noir et blanc (chat_NB.jpg). Cette fonction reçoit en entrée l'image et fournit un tableau de 256 valeurs.

A l'issue de cette fonction, voici ci-contre quelques lignes pour afficher ce tableau.

```
from PIL import Image
# Chargement de l'image
im = Image.open("chat.jpg")
# Appel de la fonction conversion en gris
image3=gris(im)
# Appel de la fonction calcul de l'histogramme
table=histogramme(image3)

# Affichage de l'histogramme
import matplotlib.pyplot as plt

plt.figure()
plt.plot(range(len(table)),table,'k')
plt.title("Histogramme")
plt.xlabel("valeur du Pixel")
plt.ylabel("Occurence")
plt.grid(True)
plt.show()
```

Une manière courante d'augmenter le contraste d'une image est «d'étendre» l'histogramme : dans l'exemple ci-dessus, on voit qu'il n'y a peu de pixels de couleur très claire ou très sombre.
Une transformation affine sur les valeurs des pixels permettrait donc de mieux utiliser l'espace des nuances de gris disponibles.

- ✓ Ecrire une fonction réalisant la fonction affine suivante. A une valeur de pixel, p fait correspondre p':
  - $0 \leq p < 30 \Rightarrow p' = 0$
  - $30 \leq p \leq 180 \Rightarrow 0 \leq p' \leq 255$ (linéairement)
  - $181 < p \leq 255 \Rightarrow p' = 255$
- ✓ Ecrire une fonction permettant d'optimiser l'image noir et blanc précédente.

Pour aller plus loin : Beaucoup de traitements d'images sont basés sur les produits de convolution. On applique spatialement sur l'image un filtre linéaire via sa matrice de convolution. Le choix des coefficients de la matrice permet de faire apparaitre des effets ou de récupérer des informations (rendre floue ou net, détecter des contours, supprimer du bruit, …).

Le principe consiste à remplacer la valeur de chaque pixel par une valeur calculée sur une petite fenêtre (3 * 3 typiquement) centrée autour de ce pixel.
Soit I une image Noir et Blanc de dimensions h×l. Le niveau d'un pixel (i,j) ∈ [0,h−1]×[0,l−1] est noté $I_{i,j}$.

On considère une matrice réelle 3×3, $F = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ appelée le filtre et deux réels d > 0 et δ appelés le diviseur et le décalage.
Le filtrage de l'image I par les données F, d et δ est l'image I' définie par $I'_{i,j}=p_{i,j} / d + \delta$
où $p_{i,j}$ est la somme, pondérée par les éléments de F, des niveaux des 9 pixels voisins de (i,j) dans I.
Plus précisément, $p_{i,j}=a.I_{i-1,j-1} + b.I_{i-1,j} + c.I_{i-1,j+1}+ d.I_{i,j-1} + e.I_{i,j} + f.I_{i,j+1}+ g.I_{i+1,j-1} + h.I_{i+1,j} + i.I_{i+1,j+1}$.

On convient que les pixels du bord de l'image ne sont pas modifiés.
De plus, le résultat $I'_{i,j}$ n'est pas toujours dans l'intervalle [0,255]. Pour se ramener à cet intervalle, deux stratégies sont possibles:

- L'*écrêtage* qui consiste à remplacer $I'_{i,j}$ par 255 (resp. 0) si $I'_{i,j}$ est > 255 (resp. < 0).
- La *normalisation*. On calcule le minimum *m* et le maximum *M* des $I'_{i,j}$ et on applique à chaque $I'_{i,j}$ l'application affine qui transforme l'intervalle [*m*,*M*] en [0,255].

- ✓ Ecrire une fonction appliquant un filtre sur une image. Cette fonction recevra l'image à modifier et un tableau de neuf valeurs. Elle renverra l'image modifiée.
- ✓ Rechercher sur internet quelques filtres et vérifier les effets sur les images à votre disposition.

$F_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ Filtre passe-bas $F_2 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$ Filtre passe-haut .....