# Représenter les images

prévision : 4/12 et 11/12

## 1 Formats vectoriel et bitmap

**Exercice 1**

1. Dans le dossier enonces/A, on a chargé deux images vectorielles gratuites fournies avec le logiciel Inkscape (format svg). On a ensuite enregistré une copie au format png. Observer le phénomène de pixellisation lorsqu'on zoome sur l'image au format png.
2. Chercher sur le web la différence entre image bitmap (ou image matricielle) et image vectorielle. Expliquer pourquoi le phénomène de pixellisation observé sur l'image png ne s'observe pas sur l'image au format svg.

## 2 Découverte du module PIL

Le module PIL permet de manipuler un fichier image (reconnaissance automatique de la largeur et de la hauteur en pixels de l'image, création d'une grille de pixels, chaque ligne de la grille correspondant à une ligne de pixels, idem pour les colonnes). A titre d'exemple, voici un script d'inversion sur une image au format pgm binaire utilisant le module PIL (cf dossier enonces/B, tester ce programme) :

**Python**

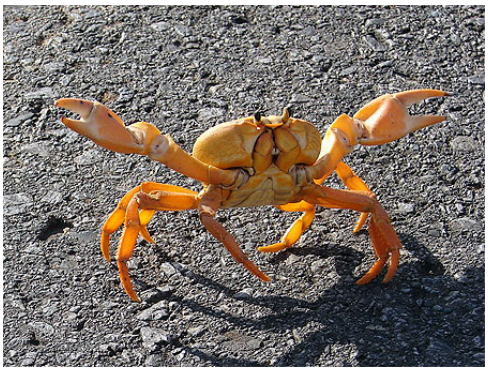
```python
# -*- coding: utf-8 -*-

from PIL import Image

# ouverture d'une image au format pgm binaire :
imageSource=Image.open("Lyceebinaire.pgm")
# sa largeur et sa hauteur en pixels :
largeur,hauteur=imageSource.size
# ouverture d'une nv image
# pour l'option "L", voir http://www.pythonware.com/library/pil/handbook/concepts.htm
# et pour Image.new(), voir http://www.pythonware.com/library/pil/handbook/image.htm
imageBut=Image.new("L",(largeur,hauteur))

# pour chaque ligne :
for y in range(hauteur):
    #pour chaque colonne :
    for x in range(largeur):
        # code du pixel (niveau de gris)
        p=imageSource.getpixel((x,y))
        # inversion du niveau de gris :
        q=255-p
        # création du pixel correspondant dans la nv image :
        imageBut.putpixel((x,y),q)


# sauvegarde de l'image créée :
imageBut.save("InversionAvecPil.pgm")
# on montre l'image :
imageBut.show()
```

On pourra explorer les possibilités à partir de la page http://www.pythonware.com/library/pil/handbook/index.htm.

# 3 Quelques transformations d'images

**Exercice 2**

1. A l'aide des outils présentés dans le paragraphe 2, transformer l'image de gauche (cf dossier enonces/D) en l'image de droite.



2. Puis en l'image suivante :

3. Puis en l'image suivante :

**Une résolution**
Première transposition :

**Python**

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

from PIL import Image
import pylab as pyl

# ouverture d'une image au format jpg :
imageSource=Image.open("crabe.jpg")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size
# création d'un tableau correspondant à l'image
# chaque ligne du tableau est une ligne de l'image
# chaque colonne du tableau est une colonne de l'image
tableauinitial=pyl.array(imageSource)
imageBut=Image.new("RGB",(hauteur,largeur))

# pour chaque ligne :
for y in range(hauteur):
    #pour chaque colonne :
    for x in range(largeur):
        # code du pixel (niveau de gris)
        p=imageSource.getpixel((x,y))
        # création du pixel correspondant dans la nv image :
        imageBut.putpixel((y,x),p)

# sauvegarde de l'image créée :
imageBut.save("transposee.jpg")
# on montre l'image :
imageBut.show()
```

Symétrie axiale :

**Python**

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

from PIL import Image
import pylab as pyl

# ouverture d'une image au format jpg :
imageSource=Image.open("crabe.jpg")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size
# création d'un tableau correspondant à l'image
# chaque ligne du tableau est une ligne de l'image
# chaque colonne du tableau est une colonne de l'image
tableauinitial=pyl.array(imageSource)
imageBut=Image.new("RGB",(largeur,hauteur))



# pour chaque ligne :
for y in range(hauteur):
    #pour chaque colonne :
    for x in range(largeur):
        # code du pixel (niveau de gris)
        p=imageSource.getpixel((x,y))
        # création du pixel correspondant dans la nv image :
        imageBut.putpixel((x,-y+hauteur-1),p)


# sauvegarde de l'image créée :
imageBut.save("sym_axe.jpg")
# on montre l'image :
imageBut.show()
```

Pour la seconde transposée, on peut transposer l'image symétrisée par la première transposition.

### Exercice 3

1. Passer de même de l'image de gauche (cf dossier enonces/E) à l'image de droite.

2. Puis essayer d'obtenir une image telle que la suivante :

**Une résolution**

Une solution pour leur faire tourner la tête :

**Python**

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

from PIL import Image
import pylab as pyl

# ouverture d'une image au format jpg :
imageSource=Image.open("cats.jpg")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size
# création d'un tableau correspondant à l'image
# chaque ligne du tableau est une ligne de l'image
# chaque colonne du tableau est une colonne de l'image
tableauinitial=pyl.array(imageSource)
imageBut=Image.new("RGB",(largeur,hauteur))

# pour chaque ligne :
for y in range(hauteur):
    #pour chaque colonne :
    for x in range(largeur):
        # code du pixel (niveau de gris)
        p=imageSource.getpixel((x,y))
        # création du pixel correspondant dans la nv image :
        imageBut.putpixel((largeur-x-1,y),p)

# sauvegarde de l'image créée :
imageBut.save("catstetetourne.jpg")
# on montre l'image :
imageBut.show()
```

Une solution pour le rougeoiment :

**Python**

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

from PIL import Image
import pylab as pyl

# ouverture d'une image au format jpg :
imageSource=Image.open("cats.jpg")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size
# création d'un tableau correspondant à l'image
# chaque ligne du tableau est une ligne de l'image
# chaque colonne du tableau est une colonne de l'image
tableauinitial=pyl.array(imageSource)
imageBut=Image.new("RGB",(largeur,hauteur))

# pour chaque ligne :
for y in range(hauteur):
    #pour chaque colonne :
    for x in range(largeur):
        # code du pixel (niveau de gris)
        p=imageSource.getpixel((x,y))
        # transfo des noirs en rouge :
        if 0<=p[0]<=90 and 0<=p[1]<=90 and 0<=p[2]<=90 : p=(255,0,0)
        # création du pixel correspondant dans la nv image :
        imageBut.putpixel((x,y),p)

# sauvegarde de l'image créée :
imageBut.save("redcat.jpg")
# on montre l'image :
imageBut.show()
```

□

### Exercice 4

Dans cet exercice, vous essaierez de passer une image en couleur (cf l'image enonces/G/fruits.jpg) en niveau de gris, donc de passer de l'image de gauche à une image telle que celle de droite :


Vous pourrez vous aider d'une lecture du document suivant : http://cache.media.eduscol.education.fr/file/ISN_Tle_S/27/3/lyceeGT_ressource_ISN_20_06_Tle_S_22_Traitement_images_1_218273.pdf.

**Une résolution**

Le document cité propose trois solutions, correspondant à trois solutions utilisées dans gimp (menu couleurs/désaturer).

Solution 1– Chaque triplet rgb (a,b,c) est remplacé par (m,m,m) où $m = \frac{1}{2}(\min(a,b,c) + \max(a,b,c))$ (cf programme ci-dessous).

Solution 2– On remplace chaque triplet $(r,g,b)$ par le triplet $(g',g',g')$ où $g' = 0.21r + 0.71v + 0.07b$.

Solution 3– On remplace chaque triplet $(r;g;b)$ par $(m;m;m)$ où $m = \frac{1}{3}(r+g+b)$. □

Avec la solution 1 :

**Python**

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

from PIL import Image
import pylab as pyl

# ouverture d'une image au format jpg :
imageSource=Image.open("fruits.jpg")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size
# création d'un tableau correspondant à l'image
# chaque ligne du tableau est une ligne de l'image
# chaque colonne du tableau est une colonne de l'image
tableauinitial=pyl.array(imageSource)
imageBut=Image.new("RGB",(largeur,hauteur))

# pour chaque ligne :
for y in range(hauteur):
    #pour chaque colonne :
    for x in range(largeur):
        # code du pixel (niveau de gris)
        p=imageSource.getpixel((x,y))
        m=min(p)
        M=max(p)
        s=(m+M)//2
        p=(s,s,s)
        # création du pixel correspondant dans la nv image :
        imageBut.putpixel((x,y),p)

# sauvegarde de l'image créée :
imageBut.save("fruitgris.jpg")
# on montre l'image :
imageBut.show()
```

On peut remarquer que le résultat du programme ci-dessus n'est pas tout à fait celui d'une image en niveaux de gris : chaque triplet est encore codé en rgb, on a donc un fichier plus lourd que nécessaire. Il faudrait envisager un changement de format.

### Exercice 5

Essayer de passer la photo de campagne du dossier enonces/H en noir et blanc.

**Une résolution**

Pour chaque pixel $(r,g,b)$, on peut calculer $m = \frac{1}{3}(r+g+b)$ puis se donner un seuil. Par exemple si $r < 120$, alors le pixel est remplacé par (0,0,0) (noir), sinon par (255,255,255) (blanc). On peut observer les différences de rendus suivant le seuil choisi. Comme une photo de campagne a beaucoup de vert, on peut aussi faire des tentatives en ne tenant compte que de la composante de vert. Voir `corriges/H/NB2.py`. □

### Exercice 6

Dans le dossier enonces/I, vous trouverez une image au format pgm binaire.
Chaque pixel est dans ce cas codé par un simple entier (entre 0 et 255) correspondant à un niveau de gris.

1. Si les entiers a,b, d, f, h, i sont tous compris entre 0 et 255, montrer que l'entier $q = (-2a - b - d + f + h + 2i)//8 + 128$ (où // permet d'obtenir le quotient de la division entière) est également compris entre 0 et 255.
2. L'objectif est maintenant de transformer l'image suivant le procédé ci-dessous.
   (a) Tout pixel de la première ligne de pixels, de la dernière ligne de pixels, de la première colonne de pixels et de la dernière colonne de pixels sera remplacé par un pixel noir.
   (b) Tout autre pixel sera codé par un niveau de gris calculé à partir des niveaux de gris de ses voisins.

   | | | |
   |---|---|---|
   | a | b | c |
   | d | e | f |
   | g | h | i |

   Le pixel central, codé dans l'image initiale par le niveau de gris e, sera codé dans l'image finale par $q = (-2a - b - d + f + h + 2i)//8 + 128$.

Ce qui a été fait ici relève d'une technique plus générale pour le travail sur les images. Vous pourrez manipuler avec le logiciel gimp :

http://docs.gimp.org/fr/plug-in-convmatrix.html.

**Une résolution**

Effet de relief. cf dossier corriges/I.

**Python**

```
# -*- coding: utf-8 -*-
from PIL import Image
import pylab as pyl

# ouverture d'une image au format jpg :
imageSource=Image.open("Lyceebinaire.pgm")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size
# création d'un tableau correspondant à l'image
# chaque ligne du tableau est une ligne de l'image
# chaque colonne du tableau est une colonne de l'image
tableauinitial=pyl.array(imageSource)
imageBut=Image.new("L",(largeur,hauteur))

for x in range(largeur):
    imageBut.putpixel((x,0),0)
    imageBut.putpixel((x,hauteur-1),0)
for y in range(hauteur):
    imageBut.putpixel((0,y),0)
    imageBut.putpixel((largeur-1,y),0)

# pour chaque ligne :
for y in range(1,hauteur-1):
    #pour chaque colonne :
    for x in range(1,largeur-1):
        # code du pixel (niveau de gris)
        a=imageSource.getpixel((x-1,y-1))
        b=imageSource.getpixel((x,y-1))
        d=imageSource.getpixel((x-1,y))
        f=imageSource.getpixel((x+1,y))
        h=imageSource.getpixel((x,y+1))
        i=imageSource.getpixel((x+1,y+1))
        q=-2*a-b-d+f+h+2*i
        q=q//8
        q+=128
        imageBut.putpixel((x,y),q)

# sauvegarde de l'image créée :
imageBut.save("convol.jpg")
# on montre l'image :
imageBut.show()
```

□

### Exercice 7

Récupérez une image au format jpg par exemple.
L'objectif est ici d'ajouter autour de l'image une bordure d'épaisseur e pixels (e étant une variable du programme). Attention, cette bordure vient se placer autour de l'image, elle n'écrase pas les pixels existants.

**Une résolution**
cf dossier corriges/J.

**Python**

```python
# -*- coding: utf-8 -*-
from PIL import Image
import pylab as pyl

bord=50
pixelbord=(255,160,0)

# ouverture d'une image au format jpg :
imageSource=Image.open("bouleau.jpg")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size
# création d'un tableau correspondant à l'image
# chaque ligne du tableau est une ligne de l'image
# chaque colonne du tableau est une colonne de l'image
tableauinitial=pyl.array(imageSource)
lf=largeur+2*bord
hf=hauteur+2*bord
imageBut=Image.new("RGB",(lf,hf))

# pour chaque ligne :
for y in range(hauteur):
    #pour chaque colonne :
    for x in range(largeur):
        # code du pixel (niveau de gris)
        p=imageSource.getpixel((x,y))
        # création du pixel correspondant dans la nv image :
        imageBut.putpixel((x+bord,y+bord),p)

for y in range(bord):
    for x in range(lf):
        imageBut.putpixel((x,y),pixelbord)

for y in range(hauteur+bord,hf):
    for x in range(lf):
        imageBut.putpixel((x,y),pixelbord)


for x in range(bord):
    for y in range(hf):
        imageBut.putpixel((x,y),pixelbord)

for x in range(largeur+bord, lf):
    for y in range(hf):
        imageBut.putpixel((x,y),pixelbord)

# sauvegarde de l'image créée :
imageBut.save("bouleaucadre.jpg")
# on montre l'image :
imageBut.show()
```

□

# 4 Utilisation d'un tableau du module pylab

*La lecture de cette section est optionnelle. Elle est là pour vous guider vers des outils supplémentaires dans le cas où vous choisiriez, dans vos projets pour le baccalauréat, de travailler sur des images.*

Dans le dossier enonces/C, on trouvera une photo de l'entrée du lycée de la Plaine de l'Ain au format jpg et un court script python à tester et comprendre.

**Python**

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

from PIL import Image
import pylab as pyl

# ouverture d'une image au format jpg :
imageSource=Image.open("lycee.jpg")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size
# création d'un tableau correspondant à l'image
# chaque ligne du tableau est une ligne de l'image
# chaque colonne du tableau est une colonne de l'image
tableau=pyl.array(imageSource)


print len(tableau)==hauteur
print len(tableau[0])==largeur
print tableau[0,0]
print tableau[hauteur-1,largeur-1]
print tableau[hauteur-1,largeur-1,0]
```

hauteur désigne la hauteur en pixels de l'image.
Les lignes du tableau sont obtenues par tableau[0], tableau[1], ..., tableau[hauteur−1].
Pour la ligne d'indice i ( i entre 0 et hauteur−1), on obtient le codage de ses pixels par tableau[i,0], tableau[i,1], ..., tableau[i,largeur−1].
Lorsqu'on entre tableau[0,0], on obtient par exemple ici [122 145 139] : c'est le codage RGB du premier pixel (en haut à gauche).
Le dernier pixel (en bas à droite) est obtenu par tableau[hauteur−1,largeur−1] (on obtient ici [ 87 105 119]).
tableau[i,j,0], tableau[i,j,1], tableau[i,j,2] sont les indices RGB du pixel de la ligne i, colonne j.

Ajouter au fichier la ligne suivante et observer :

**Python**

```
imageSource.show()
```

Remplacer maintenant cette ligne par les deux lignes suivantes :

**Python**

```
pyl.imshow(tableau)
pyl.show()
```

On lance ainsi une visualisation à partir du tableau. Des axes de repérage s'ajoutent à l'affichage. Cela permet de faciliter le travail sur l'image.
Une source : http://francoislouislaillier.developpez.com/Python/Tutoriel/InitiationNumpy/Tuto1/