Fiche

I. Image numérique

• Une **image matricielle** numérique est composée d'un ensemble de points, appelés **pixels**, disposés les uns à côté des autres. Dans le système de couleur Rouge Vert Bleu (le plus utilisé), on associe à chaque pixel un triplet d'entiers compris entre 0 et 255. Un triplet représente donc un certain mélange de rouge, de vert et de bleu. Par exemple, le triplet (255,0,0) représente le rouge et le triplet (255,255,255) représente le blanc.

• La **définition** d'une image rectangulaire est le nombre de pixels en largeur multiplié par le nombre de pixels en hauteur.

• Par exemple, cette image en noir et blanc a pour définition 30 × 20.

• Elle est donc constituée de 600 pixels.

• En ouvrant la version pbm de cette image à l'aide d'un éditeur de texte, on obtient ceci :

```
P1
# Created by GIMP version 2.10.10 PNM plug-in

30 20
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000010000
110001100000000000000000101000100001010000000000000000111000010
001100000000000000000101000110001000001010100000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000
```

• Dans cet exemple, comme il s'agit d'une image en noir et blanc, un pixel n'est pas associé à un triplet d'entiers, mais soit à l'entier 0 (pour le blanc) soit à l'entier 1 (pour le noir).

• La **résolution** d'une image est le nombre de pixels par unité de longueur.

• Considérons une image de définition 3750 × 2500. On affiche cette image à la taille de 15 cm × 10 cm. Alors la résolution de cette image est de 250 pixels par centimètre. Elle est très souvent exprimée en pixels par pouce (ppp), en sachant que 1 pouce est égal à 2,54 cm. 250 pixels par centimètre équivalent à 635 ppp.

II. Modification d'images

Nuance de gris

Voici un petit programme, en langage Python, qui permet de convertir une image en couleurs en une image en nuances de gris.

```
from PIL import Image
ASP = Image.open("logo-ASP.jpg")
largeur,hauteur=ASP.size
for y in range(hauteur):
    for x in range(largeur):
        r,v,b=ASP.getpixel((x,y))
        n=(r+v+b)//3
        ASP.putpixel((x,y),(n,n,n))
ASP.show()
ASP.save("logo-ASP-NoirEtBlanc.jpg")
```

• On importe le module Image de la bibliothèque PIL.

• On crée un objet (ASP) grâce à la méthode « open ». (Bien vérifier que l'image se trouve dans le même dossier que votre programme.)

• On récupère les dimensions de l'image (ici largeur = hauteur = 200 pixels) grâce à la méthode « size ».

• On parcourt l'ensemble des pixels de l'image à l'aide de deux boucles bornées.

• Pour chaque pixel on détermine le triplet d'entiers qui lui est associé.

• On calcule la moyenne arithmétique de ces trois entiers et on remplace l'ancien triplet par un nouveau triplet. Enfin on affiche l'image et on l'enregistre sous un nouveau nom.

• **Remarque :** Pour obtenir une nuance de gris plus propre on peut utiliser :int(round(0.299*r+ 0.587*v+ 0.114*b)) à la place de (r+v+b)//3

Augmentation de la luminosité

• Il suffit d'ajouter un entier fixe aux trois entiers associés au pixel.

• Voici un programme qui ajoute 150 aux trois canaux de couleurs (rouge, vert et bleu).

```
from PIL import Image
ASP = Image.open("logo-ASP.jpg")
largeur,hauteur=ASP.size
for y in range(hauteur):
    for x in range(largeur):
        r,v,b=ASP.getpixel((x,y))
        n_r=r+150
        n_v=v+150
        n_b=b+150
        ASP.putpixel((x,y),(n_r,n_v,n_b))
ASP.show()
ASP.save("logo-ASP-Luminosité.jpg")
```

• Pour diminuer la luminosité, il suffit, on l'aura compris, de soustraire un entier fixe aux trois entiers associés au pixel.

Négatif

• Il suffit de remplacer chaque entier a par 255-a.

Noir et blanc

• À partir d'une image en niveau de gris, on crée une image comportant uniquement du noir ou du blanc. On remplace un à un les pixels de l'image par rapport à une valeur seuil fixée (par exemple 127). De ce fait, si un pixel à une valeur supérieure au seuil (par exemple 189), il prendra la valeur 255 (blanc), et si sa valeur est inférieure (par exemple 93), il prendra la valeur 0 (noir).

III. EXIF

• Les anciens photographes professionnels possédaient un carnet papier dans lequel ils détaillaient pour chaque photo prise une liste de données. Ils précisaient le lieu, la date, l'objectif utilisé, l'ouverture...

• Les informations notées sont des **métadonnées**, c'est-à-dire des données sur la photo elle-même.

• Les **métadonnées** d'une image numérique sont des données autres que les pixels constituant l'image ; elles sont stockées dans le même fichier que l'image elle-même.

• Les données EXIF (EXIF signifie ***Exchangeable Image File***) sont utilisées par l'ensemble des appareils photo numériques.

• Sur Windows, sélectionner une photo, puis en sélectionnant « Propriétés » et en choisissant l'onglet « Détails », déterminer dans quel pays et dans quelle ville cette photo a été prise.
(Sur Mac, sélectionner une photo, puis choisir « Afficher Infos »).
On peut aussi déterminer si la photo a été prise avec un smartphone, la date et l'heure exacte de la prise de la photo...

• Ces données sont créées au moment de la création de l'image, mais elles sont modifiables par la suite grâce à certains logiciels.

• Ce petit programme en langage Python permet d'afficher les données EXIF de la photo précédente.

```
from PIL import Image
ASP = Image.open('IMG_1554.JPG')
ASPInfo = ASP._getexif()
print(ASPInfo)
```

• Il affiche alors ceci :

```
{34853: {1: 'N', 2: ((39, 1), (51, 1), (3394, 100)), 3: 'W', 4:
((4, 1), (1, 1), (5570, 100)), 7: ((12, 1), (35, 1), (1192,
100)), 12: 'K', 13: (0, 1), 16: 'T', 17: (41555, 122), 23: 'T',
24: (41555, 122), 29: '2016:12:28', 31: (175817, 99)}, 296: 2,
34665: 204, 271: 'Apple', 272: 'iPhone 4S', 305: '9.3.5', 274: 6,
306: '2016:12:28 13:35:16', 531: 1, 282: (72, 1), 283: (72, 1),
36864: b'0221', 37121: b'\x01\x02\x03\x00', 37377: (3915, 338),
36867: '2016:12:28 13:35:16', 36868: '2016:12:28 13:35:16',
37378: (4845, 1918), 37379: (8233, 757), 37380: (0, 1), 37383: 5,
37385: 16, 37386: (107, 25), 40961: 1, 40962: 3264, 41989: 35,
41990: 0, 40963: 2448, 37521: '531', 37522: '531', 37396: (1631,
1223, 881, 881), 41495: 2, 33434: (1, 3067), 33437: (12, 5),
41729: b'\x01', 34850: 2, 34855: 50, 41986: 0, 40960: b'0100',
41987: 0, 42034: ((107, 25), (107, 25), (12, 5), (12, 5)), 42035:
'Apple', 42036: 'iPhone 4S back camera 4.28mm f/2.4', 37500:
b"Apple iOS\x00\x00\x01MM\x00\x08\x00\x01\x00\t
\x00\x00\x00\x01\x00\x00\x00\x04\x00\x03\x00\x07\x00\x00\x00h
\x00\x00\x00t\x00\x04\x00\t
\x00\x00\x00\x01\x00\x00\x00\x01\x00\x05\x00\t
\x00\x00\x00\x01\x00\x00\x00\xd0\x00\x06\x00\t
\x00\x00\x00\x01\x00\x00\x00\xdb\x00\x07\x00\t
\x00\x00\x00\x01\x00\x00\x00\x01\x00\x08\x00\n
\x00\x00\x00\x03\x00\x00\x00\xdc\x00\x14\x00\t
\x00\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00\x00bplist00\xd4\x01\
x02\x03\x04\x05\x06\x07\x08YtimescaleUvalueUepochUflags\x12;\x9a
\xca\x00\x13\x00\x00\x1f\xe7\xaeYz\xad
\x10\x00\x10\x01\x08\x11\x1b!'-2;=
\x00\x00\x00\x00\x00\x00\x01\x01\x00\x00\x00\x00\x00\x00\x00\t
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00?\xff
\xff\xfe\x81\x00\x00]\x9f\xff\xff\xf1#\x00\x00\x0f!
\x00\x00\x01\x00\x00\x00\x06\xad"}
```

• On retrouve dès la première ligne les coordonnées géographiques du lieu de la prise de vue de la photo.

• On retrouve également le type de smartphone utilisé, la date de la prise de vue...

• Ce programme plus long va extraire les informations concernant les coordonnées géographiques, puis les convertir en degré décimal et accéder au site web openstreetmap.org pour afficher le lieu de la prise de vue sur une carte.

```
from PIL import Image
import webbrowser
ASP = Image.open( 'IMG_1554.JPG' )
ASPInfo = ASP._getexif()
Coord={}
Coord=ASPInfo[34853]
lat_deg=Coord[2][0][0]/Coord[2][0][1]
lat_min=Coord[2][1][0]/Coord[2][1][1]/60
lat_sec=Coord[2][2][0]/Coord[2][2][1]/3600
lat=lat_deg+lat_min+lat_sec

lon_deg=Coord[4][0][0]/Coord[4][0][1]
lon_min=Coord[4][1][0]/Coord[4][1][1]/60
lon_sec=Coord[4][2][0]/Coord[4][2][1]/3600
lon=lon_deg+lon_min+lon_sec
print("La latitude est environ égale à : ")
print(lat)
print("La longitude est environ égale à : ")
print(lon)

if Coord[3]=='W':
    lon=-lon

zoom='14'
webbrowser.open('https://www.openstreetmap.org/note/new?lat='+str(lat)+'&lon='+str(lon)+'#map='+zoom+'/'+str(lat)+'/'+str(lon))
```

• On a créé un dictionnaire (appelé « Coord ») pour récupérer les coordonnées en degrés sexagésimaux.

• Le dictionnaire contient des listes. La latitude est donnée à la clé 2 qui consiste en une liste de couples : le premier couple (degré, 1), le deuxième (minute, 1) et le troisième (seconde, 100).

IV. Création d'une image numérique par un appareil photo

• Les appareils photo numériques n'utilisent plus de pellicule, mais un **capteur photographique**.

• Il est composé de **photosites** disposés en quadrillage.

• Chacun d'eux est chapeauté par une **microlentille** qui recueille la lumière (après qu'elle a été filtrée par les filtres Infra-Rouge et Anti-Aliasing) sur toute la surface disponible et la concentre sur le photosite.

• Avant d'arriver sur le photosite, la lumière va passer dans la **matrice de Bayer**. Celle-ci sépare la lumière en laissant passer sa composante verte, rouge ou bleue. La composante verte est deux fois plus représentée dans la matrice que les autres couleurs, car l'œil humain est naturellement plus sensible au vert qu'aux autres couleurs.

• Sous la matrice de Bayer se situent les photosites, qui vont convertir la lumière reçue (donc rouge, verte ou bleue) en signal électrique, plus ou moins important suivant l'intensité de la lumière reçue.

• L'association de 4 photosites (deux ayant récoltés de la lumière verte, un ayant récolté de la lumière rouge et un ayant récolté de la lumière bleue) donnera les informations qui permettront de créer un pixel de l'image.

• L'aire d'un capteur photographique varie d'un smartphone à un appareil photo numérique et peut aller de 7,7 mm^2 à 2 169,5 mm^2. La résolution du capteur se mesure en millions de photosites.

V. Algorithmes internes

• Une fois le travail des photosites effectué, les signaux électriques arrivent au **processeur d'images**.

• Il analyse alors la couleur (et la brillance) des pixels créés et les compare avec les données des pixels voisins. Il effectue ensuite un dématriçage pour donner aux pixels une couleur appropriée et une valeur de brillance.

• D'autres algorithmes entrent alors en jeu pour la bonne gestion de la lumière et du contraste, la balance des blancs, la netteté, le débouchage des ombres, la correction automatique des distorsions ou des aberrations optiques. lls dépendent de la marque de l'appareil photo ou du smartphone.

• L'algorithme le plus connu est celui de la mise au point automatique (autofocus) par détection de contraste. Le niveau de netteté de la photo est mesuré par le contraste entre des pixels adjacents. L'algorithme modifie la mise au point jusqu'au moment où l'image devient la plus nette possible.

Exercice n°1

Exercice n°2

Exercice n°3

Exercice n°4

Exercice n°5