I. Description d'une image

1 - Fichier et image

Définition :

Une image matricielle, ou "carte de points" (de l'anglais bitmap), est une image constituée d'une matrice de points colorés. C'est-à-dire, constituée d'un tableau, d'une grille, où chaque case possède une couleur qui lui est propre et est considérée comme un point. Il s'agit donc d'une juxtaposition de points de couleurs formant, dans leur ensemble, une image.

Propriétés :

Une image matricielle est généralement définie par :
Les dimensions de sa matrice : largeur et hauteur, parfois sa taille réelle (échelle).

- La nature de ses couleurs. La plupart des images numériques utilisent les différents systèmes RVB, HLS, TLS, HUV ou quadrichromique: Cyan Magenta Yellow Black.
- Le nombre et la précision de ses couleurs/
- Des informations complémentaires telles que, typiquement : l'auteur, les spécificités techniques, une miniature, etc...

En zoomant sur une image, on s'aperçoit qu'elle est constituée d'une multitude de pixels organisés en lignes (y) et colonnes (x). Dans l'exemple ci-dessous la couleur de chaque pixel est codée sur un octet (c'est-à-dire 8 bits), et représente donc une nuance de gris parmis 255 valeurs possibles. On peut voir cette valeur comme la luminosité du pixel: la valeur 0 code la couleur noire, la valeur 255 code la couleur blanche, et les valeurs intermédiaires codent des nuances de gris de plus en plus claires à mesure que la valeur augmente.

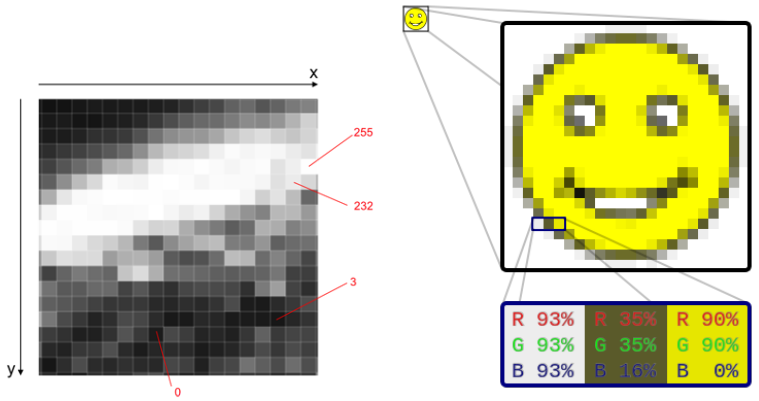

Figure 1: Description d'une figure en niveaux de gris et couleur

2 - Format de fichiers

Si toutes les images se représentent de la même façon le conteneur, c'est à dire le fichier contient plus qu'une image. Il contient dans son entête des données, plus exactement des métadonnées, qui décrivent la forme, la dimension, les traitements de compressions éventuellement subis par l'image, son auteur, la date de création et beaucoup d'autres données. Lorsqu'on parle d'une image au format jpeg, on désigne en fait un fichier avec une extension .jpeg ou .jpg, qui contient une image ayant subit une compression avec un algorithme jpeg. Les caractéristiques de la compression sont indiquées dans les métadonnées du fichier, mais pas dans la matrice qui code l'image ! Les formats les plus courants sont énoncés ci-dessous :

- jepg (Joint Photographic Expert Group): C'est un format de compression d'images avec perte de données, sans doute le plus répandu. .
- png (Portable Network Graphic) : C'est un format de compression d'images, sans perte de données, moins efficace que jpeg. Il est très utilisé sur le net car il permet de la compression en gardant une très bonne qualité d'image.
- gif (Graphics Interchange Format) : format d'image sans compression, mais avec seulement 8 bits par pixel (codage de 256 couleurs). Les images sont de taille faible. On le rencontre beaucoup sur le net.
- tiff (Tagged Image File Format) : format d'image sans compression, mais avec 16 ou 32 bits par pixel, donc des images très volumineuses.
- bmp (BitMap): le format ancestral, sans compression de données, avec des images énormes.

Figure 2: Image de 660 x 660 pixels pour 343 ko

Exemple 1 :

L'image `bouquet_rose.jpg` est une image couleur de 660×660 pixels codée sur 256 niveaux de rouge vert et bleu.

☐ 1 - Déterminer l'ordre de grandeur de la taille en octet nécessaire pour ce codage.

☐ 2 - Comparer à la taille du fichier. Conclure

II. Manipulations de base sur les images

a) Négatif

Exemple 2 :

On considère une matrice image A. Définir une fonction $negatif(A)$ renvoyant une matrice image négatif de A.

```
def negatif(image):
    l,h = len(image),len(image[0])
    for ix in range(l):
        for iy in range(h):
            image[ix][iy] = 255 - image[ix][iy]
    return An
```

Figure 3: Image en négatif

1 - Image en noir et blanc

Exemple 3 :

On considère une matrice `image`. Définir une fonction $NetB(A,seuil)$ renvoyant une matrice image en noir et blanc selon la valeur *seuil* du niveau de gris.

```
def NetB(image,seuil):
    image_NB = image.copy()
    hauteur,largeur = len(image),len(image[0])
    for i in range(hauteur):
        for j in range(largeur):
            if image[i][j] > seuil:
                image_NB[i][j] = 255
            else :
                image_NB[i][j] = 0
    return image_NB
```

Figure 4: Image en noir et blanc pour un niveau seuil de 100 et 200

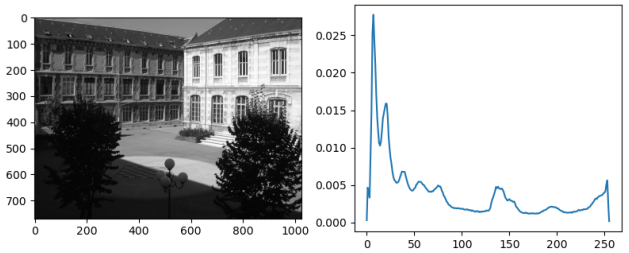

Figure 5: profil des niveaux de gris dans une image

Remarque 1 :

Le choix du seuil est important, il est souvent établi à partir du profil de niveau de gris de l'image

Exemple 4 :

Proposer un code renvoyant une liste de 256 valeurs, `L[i]` correspondant au pourcentage de pixel ayant le niveau `i`

```
def profil(image):
    L = 256*[0]
    h, l  = len(image),len(image[0])
    for i in range(h):
        for j in range(l):
            L[image[i][j]] += 1
    return [x / ( h* l ) for x in L]
```

2 - Changement de résolution

Exemple 5 :

Définir une fonction $resize(A,Nx,Ny)$ qui retourne une matrice image de taille Nx, Ny correspondant à une image réduite de A.

```
def resize(im_orig,Nx,Ny):
    l,h,t = im_orig.shape
    img = np.zeros((Nx,Ny,t), dtype=np.uint8)
    for i in range(Nx):
        for j in range(Ny):
            img [i][j] = im_orig[(i*(l-1))/Nx][(j*(h-1))/Ny]
    return img
```

Figure 6: Image réduite à 50×50

3 - Rotation d'une image

Exemple 6 :

On considère une matrice `image`. Définir une fonction`rotation(image,theta)` renvoyant une matrice image pivoté d'un angle theta à partir du pixel 0,0

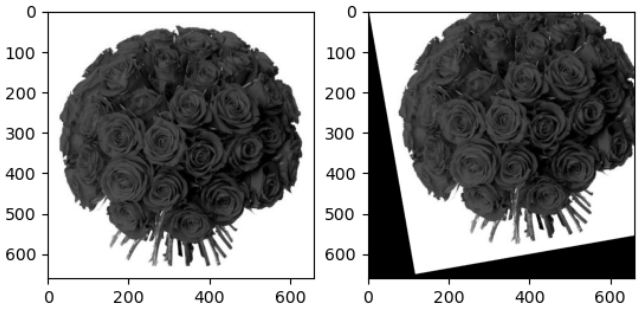

Figure 7: Image en noir et blanc pour un niveau seuil de 100 et 200

```python
def rotation(image,theta):
    theta = theta*pi/180
    h, l  = len(image),len(image[0])
    im_r = [ [0 for _ in range(l)] for _ in range(h) ]
    for i in range(h):
        for j in range(l):
            i_r = int( cos(theta)*i +sin(theta) * j )
            j_r = int( -sin(theta)*i +cos(theta) * j )
            if 0< i_r < h and 0< j_r< l :
                im_r[i][j]= image[i_r][j_r]
    return im_r
```

4 - Images en couleur

a) Changement de couleur...

Exemple 7 :

À partir du fichier `bouquet_rose.jpg`, proposer un code permettant d'obtenir un bouquet bleu en échangeant les valeurs des pixels rouge et bleu.

```python
def conversion_rouge_bleu(im_orig):
    l,h,t =  im_orig.shape
    im = np.copy(im_orig) # On fait une copie de l'original
    for i in range(l):
        for j in range(h):
            r, v, b = im_orig[i][ j]
            r,b = b,r
            im[i, j] = (r, v, b)
    return im
```

Figure 8: Image en inversant deux couleurs

b) Niveau de gris

La CIE (Commission Internationale de l'Eclairage) propose une normalisation pour les niveaux de gris, la norme 709 indique que le codage du niveau de gris vérifie la relatin :

$$0.2125 * R + 0.7154 * G + 0.0721 * B$$

Exemple 8 :

Proposer une fonction `conversion_gris(im_orig)` qui à partir de l'image d'origine sous forme de tableau (l,h,3), renvoie un tableau de taille (l,h) dont les éléments sont le niveau de gris.

Figure 9: Image en couleur et niveau de gris

Figure 10: Image floutée, puis floutée au centre uniquement

III. Manipulations par convolution

Dans toute cette partie, on n'utilisera qu'un unique tableau 2D codant une image en niveau de gris. L'utilisation de tableau 3D est possible mais évidemment plus compliquée. Vous pourrez utiliser le fichier image `Carnot.jpg` ou tout autre fichier de votre choix.

1 - Floutage

Pour flouter une image, ou au contraire en accentuer les détails, la technique la plus simple consiste à remplacer la valeur de chaque pixel par une valeur calculée sur une petite fenêtre (3×3 typiquement) centrée autour de ce pixel. Ce calcul est basée sur une matrice qui varie suivant l'effet que l'on veut obtenir.

Exemple 9 :

☐ 1 - Définir une fonction `flou(img_orig,n)` qui renvoie une image floutée par moyennage uniforme sur une fenêtre de ($n \times n$ typiquement).

☐ 2 - Pour les étudiants à l'aise, définir une fonction `flou_milieu(img_orig,pourcentage,n)` réalisant un flou sur un certain pourcentage de la partie centrale de l'image. Le flou étant obtenu par moyennage sur n pixels.

```python
def flou(img_orig,n):
    l,h = img_orig.shape
    img = np.zeros((l,h), dtype=np.uint8)
    for i in range(l-n):
        for j in range(h-n):
            img[i][j] = np.average(img_orig[i:i+n,j:j+n])
    return img


def flou_milieu(img_orig,pourcentage,n):
    l,h = img_orig.shape
    l_flou = int(pourcentage*l/100)
    h_flou = int(pourcentage*l/100)
    img = np.copy(img_orig)
    for i in range(l/2-l_flou,l/2+l_flou):
        for j in range(h/2-h_flou,h/2+h_flou):
            img[i][j] = np.average(img[i:i+n,j:j+n])
    return img
```

2 - Détection de contour

a) Gradient d'une image

Définition :

Le gradient d'une image correspond à une dérivée spatiale des niveaux de gris des pixels. Le gradient d'une fonction de deux variables est un vecteur de dimension 2 dont les coordonnées sont les dérivées selon les directions horizontale et verticale.

$$\vec{grad} f(x,y) = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

En chaque point, le gradient pointe dans la direction du plus fort changement d'intensité, et sa longueur représente le taux de variation dans cette direction. Le gradient dans une zone d'intensité constante est donc nul. Au niveau d'un contour, le gradient traverse le contour, des intensités les plus sombres aux intensités les plus claires. Les valeurs obtenues seront affectées par une valeur absolue en raison du domaine de définition des niveaux de gris variant de 0 à 255.

b) Algorithme

Exemple 10 :

Déterminer une fonction $grad(A)$ qui renvoie deux tableaux images représentant les gradients en niveau de l'image A.

Solution 1

```
1 def grad(img):
2     Nx,Ny = img.shape
3     grad_x=np.zeros((Nx,Ny))
4     grad_y=np.zeros((Nx,Ny))
5     for i in range(1,Nx):
6         for j in range(1,Ny):
7             grad_x[i][j]=abs(img[i+1,j]-img[i][j])
8             grad_y[i][j]=abs(img[i,j+1]-img[i][j])
9     return grad_x,grad_y
```

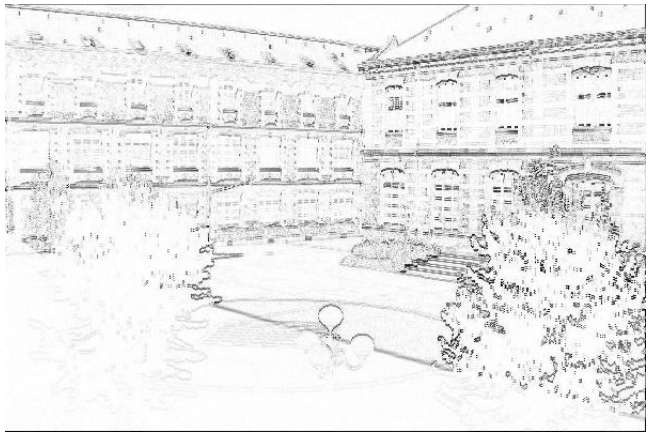

Figure 11: négatif du gradient d'image selon X

Remarque 2 :

Les contours sont alors obtenus en évaluant la norme du gradient :

$$||\vec{grad} f(x,y)|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$