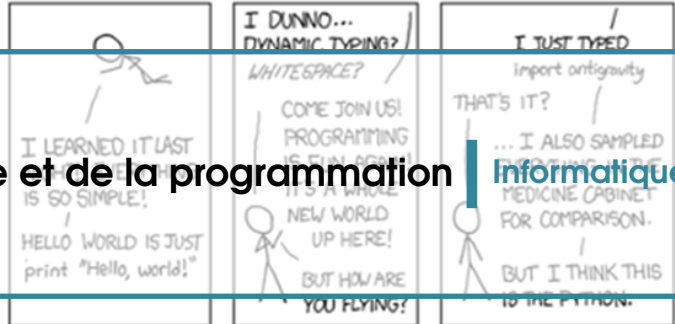
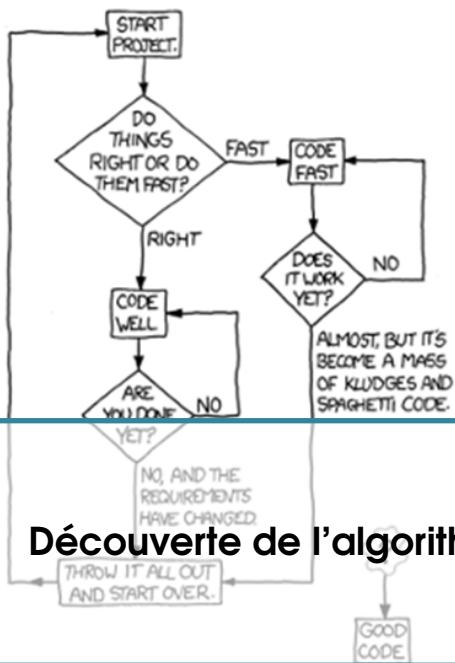


HOW TO WRITE GOOD CODE:



Découverte de l'algorithmique et de la programmation Informatique

Chapitre 3

Matrices de pixels et images

Savoirs et compétences :

- Matrices de pixels et images.


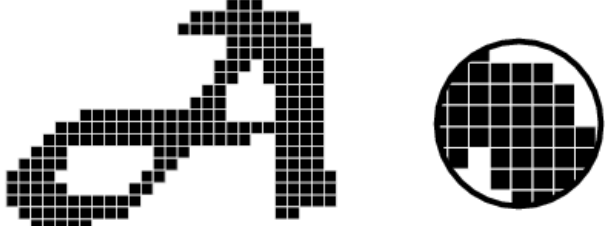
Cours

1	Codage des images	2
2	Manipulation des matrices de pixels	3
3	Transformation d'images	4
4	Modification par convolution : filtrage	5

1 Codage des images

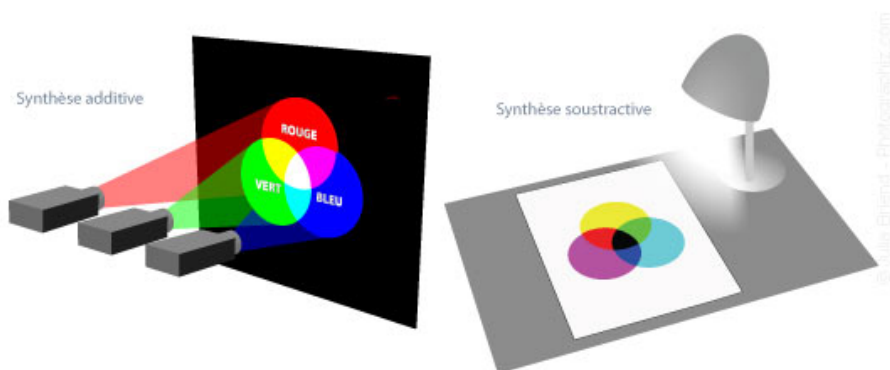
1.1 Codage numérique d'une image

Il existe deux modes de codage d'une image numérique.

Le mode vectoriel (« vector »)	Le mode matriciel (« bitmap »)
On décrit les propriétés mathématiques des formes de l'image.	Il repose sur le principe d'une grille de pixels.
	
Une image vectorielle peut être zoomée sans que cela n'altère la qualité du rendu. Ce mode est adapté aux formes et images qui ne sont pas trop complexes : logos, typographie, plans, cartes, etc. <i>Les formats de fichiers vectoriels sont PostScript, PDF, SVG, etc.</i>	Pour une image matricielle, le nombre de pixels utilisés par unité de longueur donne un rendu plus ou moins précis des formes de l'image. Un zoom trop important fait apparaître la pixellisation de l'image. <i>Les formats de fichiers matriciels sont BMP, GIF, PNG, JPEG, etc.</i>

On s'intéresse par la suite au codage des images par le mode matriciel.

1.2 Notion de colorimétrie



La **synthèse additive** est utilisée par les moniteurs et les projecteurs. Elle est constituée des trois lumières de base qui sont le Rouge, le Vert et le Bleu (RVB). Les couleurs secondaires sont plus claires que les primaires, c'est pour cela que leur synthèse est appelée additive.

La **synthèse soustractive** est utilisée dans l'imprimerie et par les imprimantes. Les couleurs primaires sont le Cyan, le Magenta, le Jaune et le Noir (CMJN). Si on mélange deux couleurs primaires, le résultat sera une couleur secondaire plus sombre qui absorbera donc plus de lumière, c'est pour cette raison que cette synthèse est nommée soustractive.

Un modèle de colorimétrie est une manière de coder des couleurs avec des nombres. Nous ne présenterons ici que le modèle RVB (Rouge Vert Bleu).

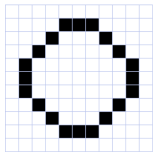
Le **modèle RVB** caractérise une couleur à l'aide de trois paramètres (chacun sur un octet, soit un nombre de 0 à 255) correspondant aux trois couleurs Rouge, Vert et Bleu.



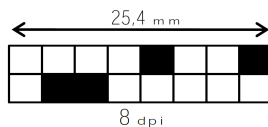
$00\ 00\ 00_{(16)} = (0,0,0) = \text{noir}$
 $FF\ FF\ FF_{(16)} = (255,255,255) = \text{blanc}$
 $FF\ 00\ 00_{(16)} = (255,0,0) = \text{rouge}$

1.3 Le mode matriciel « bitmap »

Une matrice de pixels (« picture elements ») est constituée d'un ensemble de points pour former une image. Le pixel représente ainsi le plus petit élément constitutif d'une image numérique.



La discrétisation d'une image se caractérise par sa **définition**, c'est-à-dire le nombre de pixels utilisés.



La **résolution** de l'image établit un lien entre le nombre de pixels d'une image et sa taille réelle. Elle se caractérise par un nombre de pixels par unité de longueur.

La **quantification des couleurs** est exprimée en bits par pixel (bpp) : 1 bpp (= 1 bit/pixels = deux couleurs : noir et blanc par exemple), 8 bpp (niveaux de gris entre 0 et 255 par exemple, 256 couleurs), etc.

■ **Exemple** « 800 x 600 » signifie une largeur de 800 et une hauteur de 600 pixels.

« 600 dpi » (« ppp » : point par pouce, « dpi » : dots per inch) signifie 600 pixels par pouce (1 pouce = 25,4 mm). En connaissant le nombre de pixels d'une image et la mémoire nécessaire à l'affichage d'un pixel, il est alors possible de définir exactement la taille qu'occupe le fichier.

Une définition du type 800 * 600 avec un codage sur 24 bits (3 octets) des couleurs, donne un fichier image de taille 1,44 Mo.

■

2 Manipulation des matrices de pixels

2.1 Afficher une image à partir d'un fichier (voir Activité 1)

Nous utiliserons la bibliothèque **image** de matplotlib.

```
import matplotlib.pyplot as plt # import de la bibliotheque pyplot (la même que celle utilisée ✓
    pour l'affichage de courbes)
import matplotlib.image as img # import de la bibliotheque image

im1=img.imread("image.bmp") # lecture de l'image et stockage dans la variable im1
plt.imshow(im1) # affichage de l'image
plt.show()

im1.shape #dimensions de l'image
print (im1) #affichage du contenu de la varibale im1
```

2.2 Créer une image simple (voir Activité 2)

L'instruction ci-dessous permet de créer une image vide, pour laquelle la couleur de chaque pixel sera codée sur 3 valeurs (RVB par exemple).

```
#Création d'une image vide, avec 3 valeurs par pixels
imV = np.zeros((Ly ,Lx,3)) #Ly est la dimension verticale, Lx la dimension horizontale
```

Pour modifier la valeur d'un pixel, il suffit de modifier le triplet "RVB" associé au pixel correspondant.

```
p=im2[i,j] #affecte à la variable p le contenu du pixel situé aux indices i et j
im2[i,j]=[100,50,12] #affecte au pixel situé aux indices i et j les valeurs R/G/B : 100/50/12
```

3 Transformation d'images

3.1 Symétrie d'une image

cf Activité 4

Nous voulons obtenir un effet miroir, c'est à dire une symétrie par rapport à un axe horizontal.

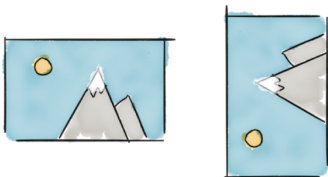


```
def symetrie(im1):
    ''' réalise une symétrie d'axe horizontal '''
    nb_lig,nb_col,nb_comp=im1.shape
    im2=np.zeros((nb_lig,nb_col,nb_comp))
    for i in range(nb_lig):
        for j in range(nb_col):
            im2[i,j]=im1[.....] # compléter ✓
            cette ligne % im2[i,j]=im1[nb_lig-i-1,✓
                               j]
    return im2
```

3.2 Rotation d'une image d'un quart de tour

cf TP

Nous souhaitons effectuer une rotation de l'image d'un quart de tour dans le sens direct.



```
def rotation_sens_direct(im1):
    nb_lig,nb_col,nb_comp=im1.shape
    im2=np.zeros(.....,nb_coul)) # ✓
    compléter cette ligne
    for i in range(nb_lig):
        for j in range(nb_col):
            im2[.....]=im1[i, j] # compléter ✓
            cette ligne
    return im2
```

3.3 Passer en niveau de gris

Une des possibilités pour obtenir une image en niveau de gris à partir d'une image codée sur les 3 composantes de couleur (bleu,vert,rouge) est de remplacer la valeur de chaque composante de chaque pixel par la moyenne des valeurs des 3 composantes du pixel considéré. Cette méthode est simple mais peu convaincante au niveau de l'image.

```
def niveau_gris(matB):
    '''convertit une image rgb en niveau de gris'''
    nb_lig,nb_col,nb_coul=matB.shape
    matC=np.zeros((nb_lig,nb_col))
    for i in range(nb_lig):
        for j in range(nb_col):
            sum=0
            for k in range(nb_coul):
                sum+=matB[i,j,k]
            moy=sum/nb_coul
            matC[i,j]=moy
    return matC
```

Pour l'affichage, il faut changer la *Colormap* de matplotlib :

```
plt.imshow ( im , cmap='gray') # affichage d'une image en niveau de gris
```

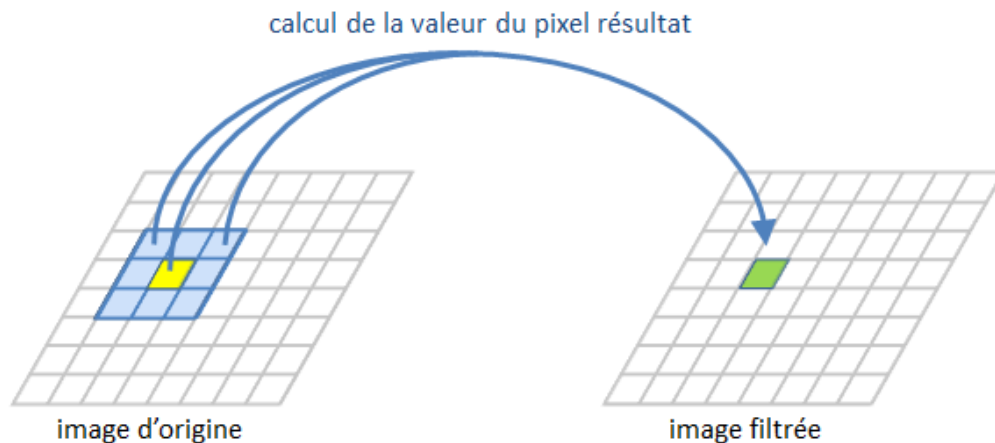
Remarque : La CIE (Commission Internationale de l'Eclairage) normalise la répartition pour obtenir un niveau de gris correct. Dans sa norme 709, il est indiqué que pour les images naturelles les poids respectifs doivent être : $0.2125 * R + 0.7154 * G + 0.0721 * B$.

4 Modification par convolution : filtrage

Le filtrage consiste à appliquer une transformation (appelée filtre) à tout ou partie d'une image numérique en appliquant un opérateur.

Un **filtre** est une transformation mathématique (appelée produit de convolution) permettant, pour chaque pixel de la zone à laquelle il s'applique, de modifier sa valeur en fonction des valeurs des pixels avoisinants, affectées de

coefficients.



Le filtre est représenté par un tableau (matrice), caractérisé par ses dimensions et ses coefficients, dont le centre correspond au pixel concerné. Les coefficients du tableau déterminent les propriétés du filtre.

■ **Exemple** Prenons le cas d'une matrice 3x3, donc un tableau de neuf nombres représentés ici par a, b, c, ..., i qui sont les paramètres du filtre et que l'on peut donc définir (cf. figure).

L'image est en niveau de gris (256 valeurs), le pixel considéré a un niveau de gris égal à 115.

La valeur 115 est remplacée par la somme de 9 produits :

$$45.a + 60.b + 81.c + 82.d + 115.e + 133.f + 130.g + 154.h + 147.i$$

Afin de normaliser ce résultat, bien souvent, on divisera ce résultat par la somme des coefficients du filtre, soit $a+b+c+d+e+f+g+h+i$. ■

	0	1	2	3	4	5
0	48	45	60	81	83	65
1	58	82	115	133	104	55
2	99	130	154	147	96	37
3	136	160	163	138	86	39
4	156	158	157	139	89	42
5	154	154	156	145	98	45

a	b	c
d	e	f
g	h	i

Extrait de l'image en niveaux de gris. Lorsqu'on applique le filtre au pixel encadré (115), le calcul fait intervenir ce même pixel et ses 8 voisins

Matrice du filtre 3x3; les valeurs a, b, c, ..., i peuvent être positives, négatives, ou nulles, entières ou non

Remarque : Qu'en est-il pour un pixel appartenant à la bordure de l'image, car il n'a pas tous ses voisins ?

La solution la plus simple est d'ignorer ces pixels : on balaye l'image de la deuxième ligne à l'avant-dernière, et pour chaque ligne, de la deuxième colonne à l'avant-dernière.

Des solutions plus évoluées utilisent aussi des matrices particulières pour les bords.

1	1	1
1	1	1
1	1	1

Flou



-1	-1	-1
-1	9	-1
-1	-1	-1

Filtre réhausseur de contraste



-2	-1	0
-1	0	1
0	1	2

Filtre d'embossage (effet de relief)



Sources :

- UPSTI

