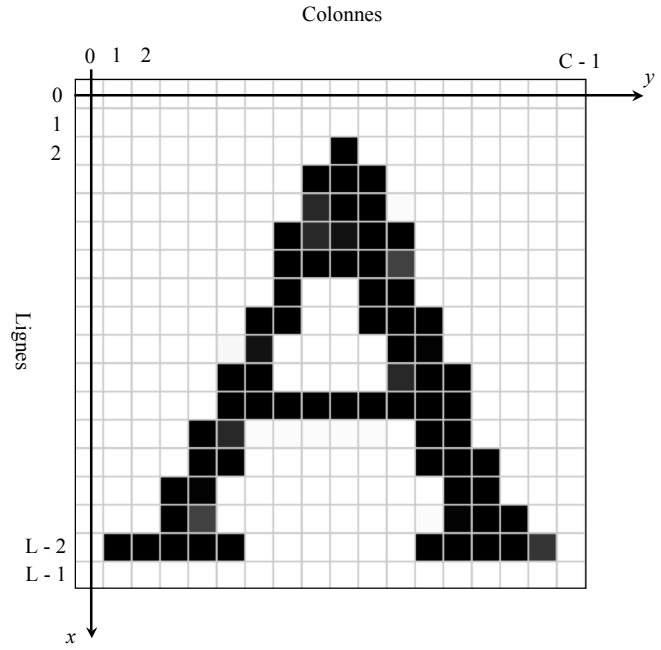
# TD : TRAITEMENT DES IMAGES

## 1 Caractéristiques d'une image

### 1.1 Les pixels

Une image est un ensemble de **pixels** qui peuvent être définis comme des zones carrées identiques de dimensions $(a \times a)$. La couleur de chaque pixel est uniforme et l'aspect non granulaire d'une image n'est dû qu'au fait que la taille des pixels est très petite. La figure ci-dessous donne un exemple d'image "binaire" représentant un A. Il n'y a ici que deux couleurs : noir et blanc.




Le repérage d'un pixel se fait dans un plan $(Oxy)$, l'axe $Ox$ étant (en général) orienté vers le bas. L'origine $O$ est au centre du pixel situé tout en haut et à gauche et Le centre de chaque pixel est alors repéré par les coordonnées $a \times (i, j)$, $i$ et $j$ étant deux entiers, avec $i \in [\![ 0, L-1 ]\!]$ et $j \in [\![ 0, C-1 ]\!]$. $L$ est le nombre de lignes de pixels et $C$ le nombre de colonnes.

### 1.2 La résolution

La résolution d'une image est donnée par la taille $a$ d'un pixel. On l'exprimer en nombre de *pixels par centimètre* ou bien en nombre de *pixels par pouce* (*inch* pour les anglo-saxons) avec la conversion : 1 pouce = 2,54 cm. Dans ce dernier cas, la résolution sera donnée en **dpi** : *dots per inch*, sachant qu'un *dot* est un point, c'est à dire un pixel.

### 1.3 La couleur

Toute couleur peut être obtenue comme une combinaison linéaire des 3 couleurs primaires rouge, vert et bleu. Une couleur peut être représentée par un triplet $(r, v, b)$ ou $r, v$, et $b$ représentent les intensités de rouge, de vert et de bleu qui donnent, par superposition, la couleur souhaitée.

Les trois **coordonnées chromatiques** $r$, $v$ et $b$ peuvent varier entre 0 et une valeur maximale $M$. Le noir correspond à (0, 0, 0) et le blanc à (M, M, M). Dans les matériels où chaque composante est un entier codé sur 1 octet (8 bits) : M = 255.

Par exemple, rgb = (217, 74, 47) représente une couleur orange. Un logiciel comme **paint** permet d'obtenir les coordonnées chromatiques de chaque couleur.

### 1.4 Représentation d'une image

Une image est donc un ensemble de pixels organisés en lignes et colonnes. Notons $L$ le nombre total de lignes de pixels numérotées de 0 à $L-1$ et $C$ le nombre de colonnes de pixels, numérotées de 0 à $C-1$.

La couleur de chaque pixel est codée sur 3 octets : r, v, b = rouge, vert et bleu. Ainsi, pour représenter une image, il faut 3 matrices $L \times C$ d'entiers naturels codés sur un octet, donc pouvant variant de 0 à 255. Ce sont les matrices **R**, **V** et **B**.

Ainsi, un pixel situé à l'intersection de la ligne $i$ et de la colonne $j$ aura trois composantes chromatiques ($R_{ij}$, $V_{ij}$, $B_{ij}$) qui sont les éléments de ces trois matrices.

Il y a différents formats de fichiers images : **.png**, **.jpg**, **.bmp**, etc... qui diffèrent selon l'organisation des données dans le fichier, la compression ou non de ces données, etc... Dans ce TD, nous allons utiliser des images au format **.png**

Vous allez utiliser les deux fonctions `getImage` et `saveImage`.

- `getImage`(*nomFichier*) prend en paramètre le nom du fichier image qui sera ici obligatoirement un fichier d'extension `.png`. Cette fonction retourne les trois matrices `R`, `V` et `B` de l'image.
- `saveImage`(*nomFichier*, `R, V, B`) qui sauvegarde une image dans le fichier *nomFichier* qui doit lui aussi nécessairement se terminer par **.png**. Il faut lui passer en paramètre les 3 matrices `R`, `V` et `B` qui représentent l'image.
- Dans tous les cas, les matrices sont des tableaux `numpy` à deux dimensions.

## 2 Quelques manipulations de base

### 2.1 Extraction des matrices R, V et B

Nous allons utiliser l'image **lena.png** ci-contre.

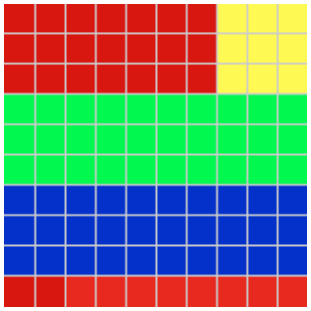
- Récupérer les 3 matrices `R`, `V` et `B` de cette image.
- Créer une matrice `Z` remplie de zéros de la même taille que `R`, `V` ou `B`.

- Enregistrer 3 images, **lenaR.png**, **lenaV.png** et **lenaB.png**. Par exemple, **lenaR.png** sera sauvegardée grâce à :

```
saveImage("lenaR.png", R, Z, Z)
```

Que constate-t-on ?

### 2.2 Écriture d'un image



Quelles sont les matrices R, V et B associées à l'image ci-contre ?

Les codes couleurs rvb utilisés sont : pour le rouge (200, 0, 0), pour le jaune (255, 255, 0), pour le vert (0, 255, 0) et pour le bleu (0, 0, 200).

Écrire un programme qui donne ces matrices et enregistrer l'image obtenue. .

### 2.3 Image en niveaux de gris (noir et blanc)

Pour obtenir une image en niveau de gris, il faut calculer la matrice :

$$G = 0{,}11 \times R + 0{,}59 \times V + 0{,}30 \times B$$

puis enregistrer l'image selon :

`saveImage("lenaG.png", G, G, G)`

Les trois matrices R, V et B sont alors identiques et valent G. Enregistrez l'image de lena en niveaux de gris. Dans la suite, c'est sur cette image que nous allons travailler.

### 2.4 Négatif de l'image

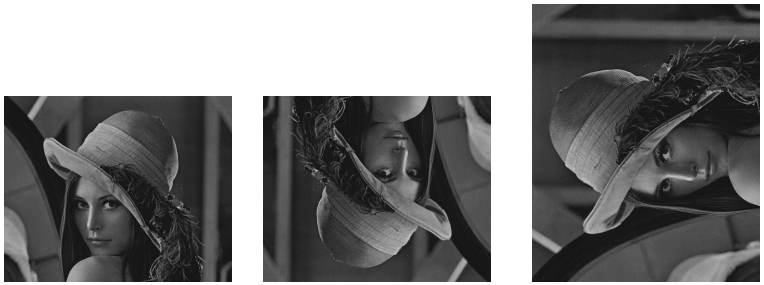
En remplaçant les éléments des matrices selon :

$$R_{ij} \to 255 - R_{ij} \quad V_{ij} \to 255 - V_{ij} \quad B_{ij} \to 255 - B_{ij}$$

Nous obtenons le négatif de l'image. Écrire un programme qui crée le négatif de l'image **lenaG.png** comme sur la figure ci-dessous

### 2.5 Retournements et symétries

Écrire un programme permettant d'obtenir les 3 images ci-contre :



## 3 Filtrage d'image

### 3.1 Quantification

On peut réduire le nombre de niveaux de gris en regroupant par exemple tous les niveaux chromatique entre 0 et 127 en un seul niveau 0, et les niveaux entre 128 et 255 en un seul niveau 255. On obtient :

On peut être plus "fin", en regroupant les niveaux chromatiques en 4, ou 8 ou 16 niveaux. Essayer.

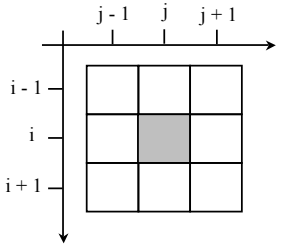
### 3.2 Traitement par groupe de pixels

#### 3.2.1 Définition

Dans les opérations précédentes, chaque pixel n'est modifié que selon sa propre valeur chromatique. Un filtrage par groupe consiste à

modifier chaque pixel selon sa valeur et celle des pixels voisins. Nous nous limiterons aux 8 pixels entourant un pixel central.

Un pixel repéré par $(i, j)$ est entouré des 8 pixels $(i-1, j-1)$, $(i, j-1)$, $(i+1, j-1)$, $(i, j-1)$, $(i, j+1)$, $(i+1, j-1)$, $(i+1, j)$ et $(i+1, j+1)$.




Nous allons définir une matrice de transformation :

$$T = \begin{pmatrix} t_1 & t_2 & t_3 \\ t_4 & t_5 & t_6 \\ t_7 & t_8 & t_9 \end{pmatrix}$$

de sorte que, si `M` désigne l'une des 3 matrices `R`, `V` ou `B`, elle soit transformée en une nouvelle matrice `M'` telle que :

$$\begin{aligned} M_{ij} \longrightarrow M'_{ij} &= t_1\, M_{i-1,j-1} + t_2\, M_{i-1,j} + t_3\, M_{i-1,j+1} \\ &+ t_4\, M_{i1,j-1} + t_5\, M_{i,j} + t_6\, M_{i,j+1} \\ &+ t_7\, M_{i+1,j-1} + t_8\, M_{i+1,j} + t_9\, M_{i+1,j+1} \end{aligned}$$

### 3.2.2 Filtre de lissage ou floutage

Reprenons l'image en niveaux de gris **lenaG.png**. Lui appliquer la transformation définie par :

$$T = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Vous pourrez recommencer cette opération un ou deux fois de plus. Qu'obtient-on ? Expliquer le résultat.

## 3.3 Détection des contours

Pour sélectionner des objets sur une image, on peut essayer de détecter leurs bords, i.e. des zones de brusque changement de niveaux de gris. On remplace alors un pixel par une mesure des écarts des voisins immédiats donnée par exemple par :

$$M'_{ij} = \sqrt{(M_{i,j+1} - M_{i,j-1})^2 + (M_{i+1,j} - M_{i-1,j})^2}$$

Appliquer cette transformation à **lenaG.png**. Expliquer le résultat. Comment obtenir alors l'image ci-dessous ?