# La simulation numérique

## Traitement d'images

Anis SAIED

Insitut Préparatoire aux Etudes d'Ingénieurs de Nabeul
.:: UNIVERSITE DE CARTHAGE ::.

AU 2016-2017

https://cahier-de-prepa.fr/info-ipein/
anis_saied@hotmail.com

# Introduction

- Une image numérique ...

# Introduction

- Une image numérique ...
- Une image peut être en couleurs, en noir et blanc, en niveaux de gris...

# Introduction

- Une image numérique ...
- Une image peut être en couleurs, en noir et blanc, en niveaux de gris...
- Une image peut être 2D, 3D (ex : IRM), 3D + Temps (film)

# Introduction

- Une image numérique ...
- Une image peut être en couleurs, en noir et blanc, en niveaux de gris...
- Une image peut être 2D, 3D (ex : IRM), 3D + Temps (film)
- Nous intéresserons durant notre cours uniquement au cas des images 2D enregistrés sur un support numérique et nous verrons comment les couleurs sont traités.

# Introduction

- Le but de ce TP est de réaliser les différentes fonctions de traitement d'image.

# Introduction

- Le but de ce TP est de réaliser les différentes fonctions de traitement d'image.
- Deux types d'images numériques :

# Introduction

- Le but de ce TP est de réaliser les différentes fonctions de traitement d'image.
- Deux types d'images numériques :
  1. **Image vectorielle :**
     Les données sont représentées par des formes géométriques dont on précise les propriétés mathématiques.
     Exemples : droite représentée par deux points (x1, y1) et (x2, y2), cercle représenté par son centre (x3, y3) et son rayon R, ....
     Ce type de représentation permet l'agrandissement d'images sans perte de qualité.

# Introduction

- Le but de ce TP est de réaliser les différentes fonctions de traitement d'image.
- Deux types d'images numériques :
  1. **Image vectorielle :**
     Les données sont représentées par des formes géométriques dont on précise les propriétés mathématiques.
     Exemples : droite représentée par deux points (x1, y1) et (x2, y2), cercle représenté par son centre (x3, y3) et son rayon R, ....
     Ce type de représentation permet l'agrandissement d'images sans perte de qualité.
  2. **Image matricielle :**
     Une image matricielle est représentée par une matrice de pixels.

# Introduction

- **Image matricielle :**
  Une image matricielle est représentée par une matrice de pixels.

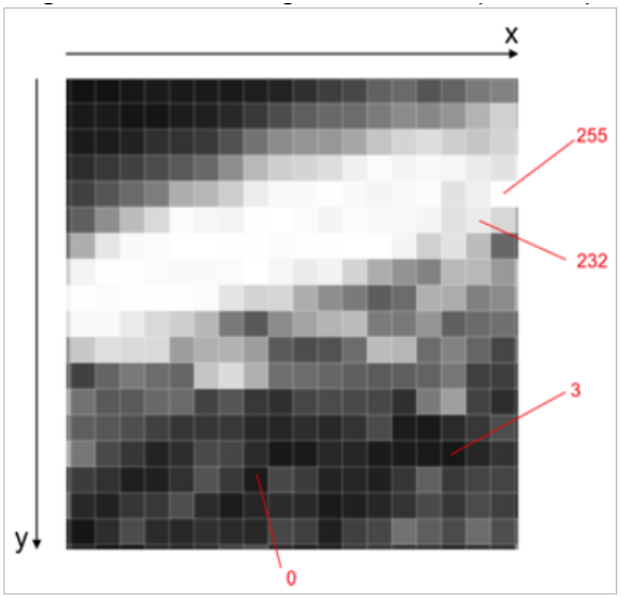

Image en niveaux de gris zoomée

x : largeur de l'image ou son nombre de colonnes
y : hauteur de l'image ou son nombre de lignes

# Introduction

- pixel = picture element ;

# Introduction

- pixel = picture element ;
- C'est le plus petit élement d'une image ;

# Introduction

- pixel = picture element ;
- C'est le plus petit élement d'une image ;
- Un pixel contient des informations sur une zone de l'espace numérisé ;

# Introduction

- pixel = picture element ;
- C'est le plus petit élement d'une image ;
- Un pixel contient des informations sur une zone de l'espace numérisé ;
- Ces informations sont enregistrés sous la forme des nombre entiers ;

# Introduction

- pixel = picture element ;
- C'est le plus petit élement d'une image ;
- Un pixel contient des informations sur une zone de l'espace numérisé ;
- Ces informations sont enregistrés sous la forme des nombre entiers ;
- Chaque pixel code une couleur dont la valeur est codée sur un nombre de bits (1 bit = image noir et blanc, 8 bits = image en niveaux de gris ,24 bits = image en couleurs) ;

# Introduction

- pixel = picture element ;
- C'est le plus petit élement d'une image ;
- Un pixel contient des informations sur une zone de l'espace numérisé ;
- Ces informations sont enregistrés sous la forme des nombre entiers ;
- Chaque pixel code une couleur dont la valeur est codée sur un nombre de bits (1 bit = image noir et blanc, 8 bits = image en niveaux de gris ,24 bits = image en couleurs) ;
- Ce qui définit la **quantification des couleurs** (nombre de couleurs dans l'image) ou le **mode de l'image**.

# Mode d'image

Le mode de l'image est exprimé en nombre de bits par pixel (bpp) codant la couleur du pixel :

- **Mode 1 bit** ou **mode Noir et Blanc** à deux couleurs (valeur 0 : Noir et valeur 1 : Blanc). C'est le **mode** « **1** » en Python

# Mode d'image

Le mode de l'image est exprimé en nombre de bits par pixel (bpp) codant la couleur du pixel :

- **Mode 1 bit** ou **mode Noir et Blanc** à deux couleurs (valeur 0 : Noir et valeur 1 : Blanc). C'est le **mode** ≪ **1** ≫ en Python
- **Mode 8 bits** ou **mode Niveaux de gris à 256 couleurs** où la valeur (luminosité) de chaque pixel est comprise entre 0 et 255 (avec 0 : Noir et 255 : Blanc) et codée sur un octet. C'est le **mode** ≪ **L** ≫ en Python
  Exemple : 00000000 code le Noir

# Mode d'image

Le mode de l'image est exprimé en nombre de bits par pixel (bpp) codant la couleur du pixel :

- **Mode 1 bit** ou **mode Noir et Blanc** à deux couleurs (valeur 0 : Noir et valeur 1 : Blanc). C'est le **mode** ≪ **1** ≫ en Python
- **Mode 8 bits** ou **mode Niveaux de gris à 256 couleurs** où la valeur (luminosité) de chaque pixel est comprise entre 0 et 255 (avec 0 : Noir et 255 : Blanc) et codée sur un octet. C'est le **mode** ≪ **L** ≫ en Python
  Exemple : 00000000 code le Noir
- **Mode 24 bits** ou **mode RVB** où chaque couleur de pixel est un mélange de trois teintes rouge, vert et bleu. La valeur de chaque teinte est comprise entre 0 et 255 et codée sur un octet. C'est le **mode** ≪ **RGB** ≫ en Python.
  Exemple : (109, 148, 114)

# Mode d'image

A partir des dimensions de l'image (largeur et hauteur) et du mode de l'image, on peut déduire sa définition et sa résolution :

- **Définition** ou **taille du fichier image** est le nombre de pixels utilisés.
  taille du fichier image = largeur × hauteur × nombre de bits de couleur par pixel.
  Exemple : pour une image de 800 pixels de largeur et de 600 pixels de hauteur avec chaque pixel codé sur 24 bits (3 octets) de couleur, son fichier occupe 1,44 Mo.

# Mode d'image

A partir des dimensions de l'image (largeur et hauteur) et du mode de l'image, on peut déduire sa définition et sa résolution :

- **Définition** ou **taille du fichier image** est le nombre de pixels utilisés.
  taille du fichier image = largeur × hauteur × nombre de bits de couleur par pixel.
  Exemple : pour une image de 800 pixels de largeur et de 600 pixels de hauteur avec chaque pixel codé sur 24 bits (3 octets) de couleur, son fichier occupe 1,44 Mo.
- La **Résolution** est le nombre de pixels par unité de longueur, exprimé en ppp (pixel par pouce avec 1 pouce = 25,4 mm).
  La résolution définit le degré de détail.

# Module Python de traitement d'image

- Il existe plusieurs modules permettant de manipuler une image sous `Python`.

# Module Python de traitement d'image

- Il existe plusieurs modules permettant de manipuler une image sous `Python`.
- Nous utiliserons le module `PIL`(Python Imaging Library).

# Module Python de traitement d'image

- Il existe plusieurs modules permettant de manipuler une image sous `Python`.
- Nous utiliserons le module `PIL`(Python Imaging Library).
- `PIL` permet d'ouvrir/créer des fichiers images et de les manipuler (accès à la valeur d'un pixel, modification d'un pixel, etc.)

```
from PIL.Image import *
import PIL.Image
import PIL.Image as im
```

# Ouverture et affichage d'une image

```
1 from PIL.Image import *
2
3 # Ouverture d'un fichier image et creation d'un objet Image
4 tiger=open('tiger.jpg')
5
6 #Affichage de l'image dans une fenetre
7 tiger.show()
```

# Ouverture et affichage d'une image

```
1  # Retour d'une sequence de valeurs des pixels
2  data = tiger.getdata()
3  l = list(data)
4
5  print(l)
6  # [(90, 84, 96), (80, 73, 81), ... , (240, 231, 172)]
7
8  print(len(l)) #94208
9
10 # Recuperer la largeur et l'hauteur (taille) de l'image
11 Largeur, hauteur=tiger.size ;
12
13 print(Largeur,hauteur) #368 256 ; 368 * 256 = 94208
```

# Modifier mode d'image

```
1 # Retour une copie convertie d'image dans un mode different
2 Img_gris=tiger.convert('L') # Mode 8 bits, valeur de 0 a 255
3
4 #Affichage de l'image dans une fenetre
5 Img_gris.show()
6
7 #Sauvegarde de l'image dans un fichier
8 Img_gris.save('tigergris.bmp')
```

# Création d'une image

```
1 #Creation d'un nouvel objet Image en precisant
2 #le mode et les dimensions
3 nouveau=new('RGB', (Largeur, hauteur))
4
5 #Affichage de l'image dans une fenetre
6 nouveau.show()
```

- Par défaut, tous les pixels sont mis à 0. l'image est noire.
- Il est ensuite possible de modifier les valeurs des pixels pour créer une image en couleurs .

# Remplissage d'une image

```
1 #Remplissage d'une image a partir d'une liste de pixels
2 nouveau.putdata(list(data))
3 nouveau.show()
```

# Modifier une image

```
1  #Retour d'un tableau de pixels
2  Tabpixel=Img_gris.load()
3  #Recuperation de la valeur d'un pixel a une position donnee
4  p1=Img_gris.getpixel((0,1)) # == Tabpixel[0,1]
5  print(p1) #107
6  #Modifier la valeur d'un pixel a une position donnee
7  Img_gris.putpixel((0,1),255) # en RGB : ((0,1),(r,g,b))
8  # ou Tabpixel[0,1] = 255
9  p1=Img_gris.getpixel((0,1))
10 print(p1) #255
11 Img_gris.show()
```

# Modifier une image

- Avant d'effectuer un traitement sur l'image, il faut la décomposer en pixels.
- Avec le module `numpy`, on transforme directement l'image en tableau.

```
1  import numpy
2  import PIL.image
3
4  #Retour d'un tableau de pixels
5  Tabpixel = numpy.array(tiger)
6
7  #Exemple d'un traitement simple
8  for lignes in Tabpixel:
9          for p in lignes :
10                 p[0],p[1] = p[1],p[0]
11
12 #Creer une image a partir d'un tableau de pixels
13 nouveau = Image.fromarray(Tabpixel)
14 nouveau.show()
```

# Copier et redimensionner une image

```
1 # Copie d'une image
2 tiger1=tiger.copy()
3
4 #Creation d'une copie redimensionnee
5 tiger1.resize((120,120)).show()
```

# copier modifier une image

```
1 # Creation d'une copie transposee (retournee)
2
3 Img_gris.transpose(FLIP_LEFT_RIGHT).show()
```

# Copier et modifier une image

```
1 #Creation d'une copie retournee de l'image originale d'un
2 # angle en degres a partir de son centre dans le sens
3 # contraire aux aiguilles d'une montre
4
5 Img_gris.rotate(45).show()
```

# Exercices 1

1. Utiliser le module `PIL` pour inverser les niveaux de gris d'une image
2. Ajouter autour de l'image une bordure d'épaisseur **e** pixels. Attention, cette bordure vient se placer autour de l'image, elle n'écrase pas les pixels existants.
3. Créer une image carrée (par exemple de taille 100 sur 100) ayant la forme d'un **X** *noir* sur fond *blanc*, le **X** joignant les coins du carré.