## Université Echahid Hamma Lakhder El Oued.

# **Module :** Image, son, vidéo codage et transmission

## TP2

## L'image couleur

### Objectifs du TP2 :

Le but de ce TP est de réaliser des manipulations sur l'image couleur sous MATLAB. En effet, la plus part des ordinateurs utilisent des profondeurs différentes pour chaque pixel de l'image (8, 16, 24 bit) ; ce nombre détermine le nombre de couleurs qui peuvent être affichées. Le 24 bit utilise 8 bit pour chaque primaire (RVB). Ce mode offre alors 16 777 216 couleurs distinctes.
Afin de faciliter la gestion informatique des images, une tendance consiste à coder le contenu de l'image par l'intermédiaire d'une table des couleurs (Look Up Table). Ce codage implique que le nombre de niveaux codés est plus petit que ce que nous pourrions faire avec de la vraie couleur. L'idée est de ne considérer qu'une seule matrice dont le contenu se résume à une succession d'adresses dans une table nommée palette (colormap). Généralement le nombre de niveaux codés correspond à une table à 256 adresses.

Rappelons que : Image = matrice de pixel X.
L'accès à un élément pixel est par X(i,j), i = ième ligne, j = jème colonne.

### 1. Cas d'une image avec des vraies couleurs

Nous allons lire l'image couleur **autumn.tif**.
**img=imread('autumn.tif');**

Visualisez le contenu de la variable img. Que remarquez-vous ?

Cette image n'est pas une matrice mais un ensemble de trois matrices, nous l'affirmons par :
**size(img)**

Cette commande indique dans son premier chiffre le nombre de ligne, dans son deuxième chiffre le nombre de colonnes et dans son troisième chiffre 3 s'il y a effectivement trois composantes couleurs pour coder la couleur de chaque pixel. En effet, toute couleur peut être obtenue en mélangeant une certaine quantité (= intensité) de Rouge, une certaine quantité de Vert, et une certaine quantité de Bleu.

Nous pouvons visualiser en niveaux de gris la composante rouge de l'image couleur :
**figure(1); imshow(img(:,:,1))**

Nous pouvons visualiser en niveaux de gris la composante verte de l'image couleur :
**figure(2); imshow(img(:,:,2))**

Nous pouvons visualiser en niveaux de gris la composante bleue de l'image couleur :
**figure(3); imshow(img(:,:,3))**

Nous pouvons visualiser l'ensemble des trois matrices sous la forme d'une image couleur :
**figure(4); imshow(img);**

Nous allons transformer l'image **autumn.tif** en niveau de gris :
**imggr = rgb2gray(img) ;**
**imshow(imggr) ;**

Nous allons transformer l'image **autumn.tif** en binaire :
**imgbnr = im2bw(img) ;**
**imshow(imgbnr) ;**

Rappelons que le traitement d'image amène à faire des opérations sur les valeurs de chaque pixel. Il faut donc que ces valeurs soient un type adapté appelé **double** en Matlab. La convention est alors que les valeurs doivent être comprises entre 0 et 1.

**imgbis=double(img)/255;**

Pour vérifier si une image est au format entier entre 0 et 255 (uint8) ou au format double (double), nous utilisons la commande suivante :

**class(img)**
**class(imgbis)**

Nous pouvons vérifier les valeurs prises par les images avec :
**max(max(max(img)))**
**min(min(min(imgbis)))**

Nous allons à présent construire une image couleur :
Pour représenter la couleur d'un pixel il faut donc donner 3 nombres, qui correspondent au dosage de 3 couleurs de base : Rouge, Vert et Bleu (en anglais, RGB : Red, Green, Blue). Nous pouvons ainsi représenter une image couleur par trois matrices, chaque matrice correspondant à une couleur de base :

**R = [0.5 0 1; 0 1 0; 1 0 0.5];**
**G = [0.5 0.6 0; 0.6 0 0.6; 0 0.6 0.5];**
**B = [0.5 1 0; 1 0 1; 0 1 0.5];**
**RGB = cat(3,R,G,B);**
**imshow(RGB)**
**imshow(RGB,'notruesize');**

Visualisez le contenu de la variable RGB.
Si nous prenons par exemple le pixel situé au centre, on constate que le dosage des trois couleurs de base est (r=1, g=0, b=0), ce qui donne du rouge. De la même façon, pour le pixel situé en haut à gauche, on a (r=g=b=0.5) ce qui donne du gris.

Nous pouvons également utiliser les commandes suivantes pour réaliser une image rouge et bleue :
**im=zeros(10,10,3);**
**im(:,:,1)=[ones(10,5) zeros(10,5)];**
**im(:,:,3)=[zeros(10,5) ones(10,5)];**
**figure(1); imshow(im);**

Nous allons afficher la composante verte du pixel de coordonnées (2,2) de l'image RGB :
**RGB(2,2,2)**

Nous allons afficher le triplet couleur du pixel de coordonnées (2,2)
**RGB(2,2, :)**

## 2. Cas d'une image couleur indexée

Pour diminuer la taille informatique des images couleurs, le nombre des couleurs accessibles est limité à une palette (256 couleurs RGB par exemple). L'information de chaque pixel (un entier sur 8 ou 16 bits) renvoie à l'une des couleurs de la palette (map); ce contenu du pixel est donc le numéro d'index de la couleur sur la palette. La palette (map) est un élément essentiel de l'information puisque sans elle, l'image n'est qu'un tableau de nombre sans relation avec une couleur ou un niveau de gris. Il est donc indispensable de récupérer la palette (map) lors de la lecture d'un fichier image :

```
[image-ind,map]=imread('forest.tif');
imshow(image-ind,map);
```

Cette image couleur est manifestement stockée sous la forme d'une table d'indice qui renvoie à une table de couleurs parce que : **image-ind** est une matrice dont les composantes ne sont pas des teintes. En effet si nous cherchons à les afficher nous obtiendrons une image qui n'a pas de sens.
Visualisez le contenu de : **image-ind** et **map**.

Nous allons afficher la première ligne de map pour voir à quelle teinte ou couleur elle référence :
**map(1,:)**

Nous allons à présent construire une image couleur indexée :

```
F= [1 2 3 ; 2 3 2 ; 3 2 1] ;
mapp=[0.5 0.5 0.5 ; 0 0.6 1 ; 1 0 0] ;
Imshow(F, mapp, 'notruesize') ;
```

Nous pouvons convertir une image couleur indexée à un triplet de matrice correspondant au rouge, vert et bleu :

```
F2=ind2rgb(F,mapp);
```

Nous pouvons également convertir une image couleur indexée à une image en niveau de gris :

```
F3=ind2gray(F,mapp);
imshow(F3, 'notruesize');
```

Nous allons démontrer comment plusieurs images peuvent être affichées sur une seule figure grâce à la fonction **subplot** :

```
load clown;
[R, G, B] = ind2rgb (X, Map);
subplot(2,3,2); imshow(map);
subplot (2,3,4); imshow(R, 256);
subplot(2,3,5); imshow(G, 256);
subplot(2,3,6); imshow(B, 256);
```