Exercice Chapitre 2 – Outils de base du traitement d'images - LUT

Cet exercice est essentiellement un exercice d'observation (peu de programmation de votre part) durant lequel vous allez non seulement vous familiariser avec l'utilisation de Matlab pour le traitement d'image (ce qui vous aidera pour les pratiques suivantes) mais aussi apprendre à utiliser les outils de bases du traitement d'image.

Lancez Matlab et mettez à jour la liste des chemins dans le path browser.

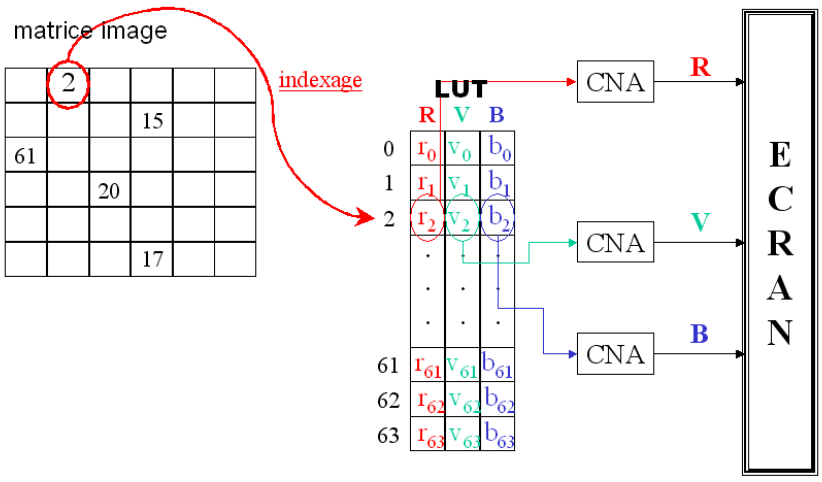
Création de LUT (*Look Up Table)*

1 – Début de session
À partir de la console Matlab, ouvrez un nouveau fichier « M-File » dans lequel vous allez saisir vos commandes et dont chaque ligne (se terminant par un « ; ») sera ensuite interprétée.

2 – Ouverture d'une image
Commencez par charger l'image *CLOWN_LUMI.BMP* en niveaux de gris avec ***imread*** (jetez un coup d'œil à l'aide). Observez le type des données.

3 – Affichage et trans-typage
L'affichage d'une image peut se faire avec les fonctions ***image***, ***imagesc*** et ***imshow***. Observez les différences, faites également un essai en trans-typant vos données.
(cf. Exercice « *Prise en main Matlab* » du chapitre 1 : essayez *image* = ***double***(*image*))

4 – Exemples de LUT
Pour l'affichage d'une image monochrome (tableau 2-D) Matlab utilise une LUT par défaut qui associe à chaque élément de la matrice image une couleur. Cette LUT par défaut présente 64 couleurs différentes en sortie (commande ***colormap*** pour afficher les niveaux de la LUT par défaut). Le fonctionnement est décrit sur la figure ci dessous.



Les différentes valeurs « i » de la matrice image vont être interprétées comme des index de la LUT par défaut. À chaque index « i » va correspondre une couleur créée par la combinaison r_i, v_i et b_i des couleurs primaires Rouge-Vert-Bleu. En effet, les valeurs r_i, v_i et b_i sont envoyées dans un Convertisseur Numérique Analogique (CNA) de façon à venir exciter respectivement les canons Rouge, Vert, et Bleu de l'écran. On obtient alors à l'affichage une couleur créée par la synthèse additive de trois couleurs primaires.

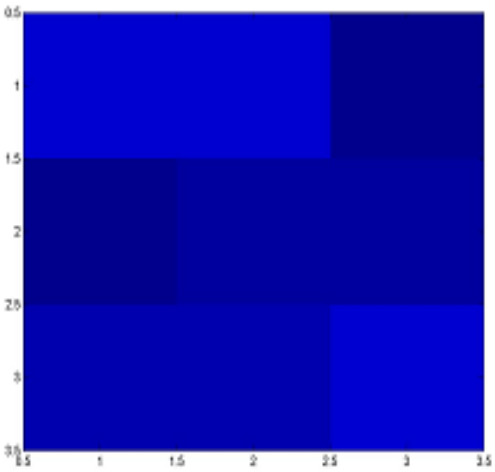
Sous Matlab, il est possible de créer ses propres LUT et de les appliquer à l'aide de la fonction ***colormap***. Les valeurs doivent cependant être normalisées sur l'intervalle [0, 1].
Prenons, par exemple, le cas simple d'une matrice M de taille 3×3 :

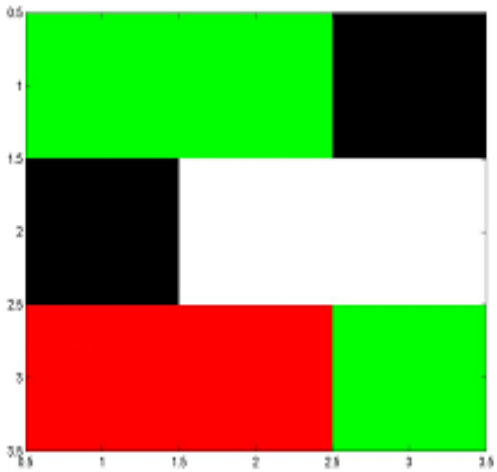
$$\begin{bmatrix} 5 & 5 & 1 \\ 1 & 2 & 2 \\ 3 & 3 & 5 \end{bmatrix}$$

Cette matrice comprend quatre valeurs distinctes. On se propose d'afficher : « 1 » en noir, « 2 » en blanc, « 3 » en rouge, et « 5 » en vert. Pour cela on crée une LUT « map4C » dont les sorties sont les quatre couleurs souhaitées. Pour appliquer cette LUT à l'image affichée, on tape la commande ***colormap(map4C)*** :

```
M=[5 5 1;1 2 2;3 3 5] ;
% Définition de la LUT
r=[0 1 1 0];
v=[0 1 0 1];
b=[0 1 0 0];
map4C=[r' v' b'];
image(M)
colormap(map4C)
```

Résultats avant et après l'utilisation de la commande ***colormap()*** :

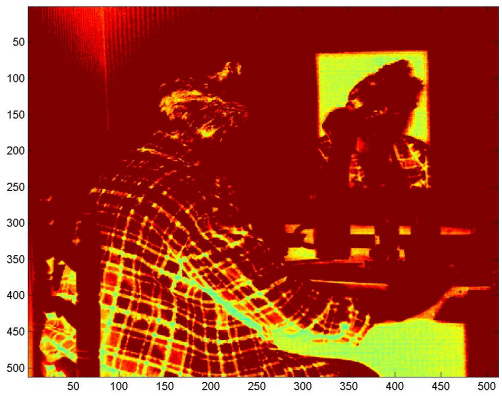




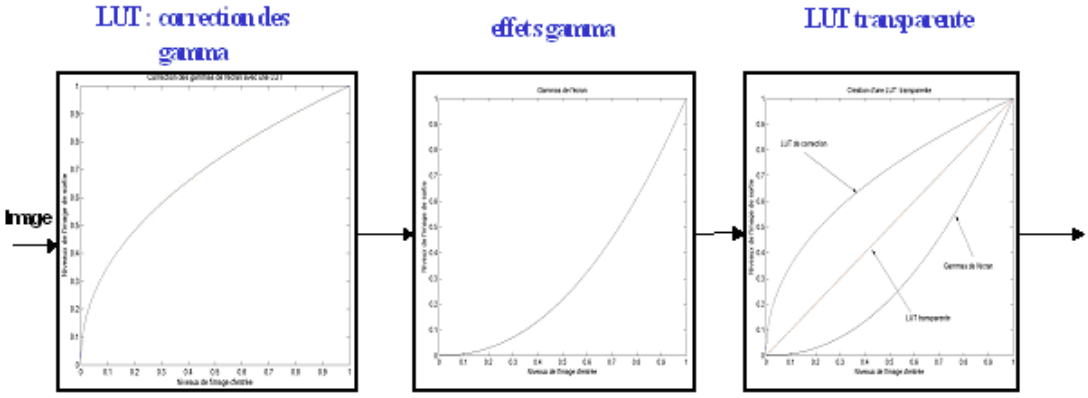
Pour afficher une image monochrome en niveau de gris, il faut donc utiliser une LUT dont les couleurs en sortie ne sont que des nuances de gris.
Dans ce cas : $\forall i \in [0, 255], r_i = v_i = b_i$ (généralement la valeur de luminance d'un pixel d'une image monochrome est codée sur 8 bits, donc 256 niveaux d'intensité possibles).

ACTION : Rapatriez le script ***lutndg.m*** dans votre répertoire de travail. Ce script sert à créer une LUT qui permet d'afficher une image monochrome en niveaux de gris (ndg). Après avoir ouvert et analysé ce fichier script, affichez une image monochrome de votre choix et tapez la commande ***lutndg.m***.

Voici les images avant et après utilisation de la LUT contenue dans le script ***lutndg.m*** :



On propose un autre exemple de création de LUT sous Matlab : un écran d'affichage génère des « effets gamma » qu'il est possible de compenser (transformation non-linéaire des niveaux de l'image d'entrée). Pour cela, on crée la LUT inverse à celle des effets gamma de l'écran afin que la composée « LUT correction o effets gamma » constitue une LUT transparente.

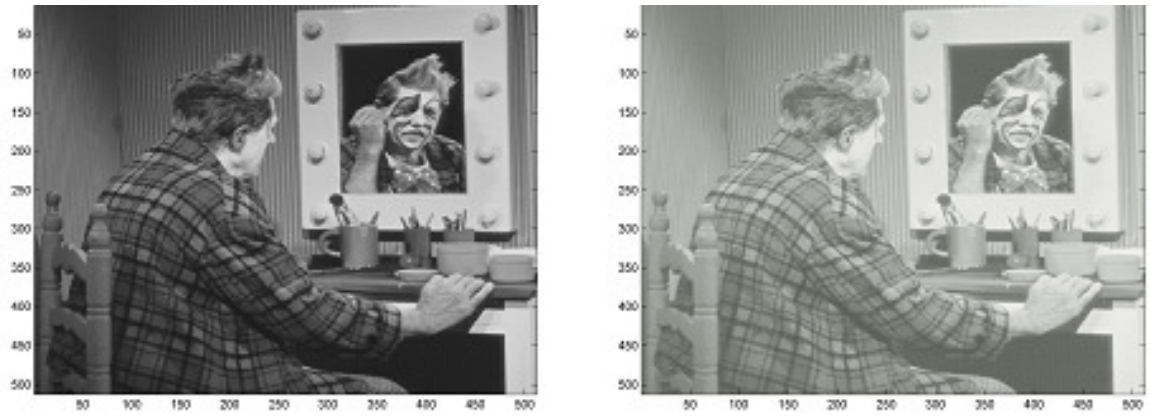


Voici un exemple de script pour comparer l'image avant et après compensation du gamma de l'écran :

```
I = imread('CLOWN_LUMI.BMP');
% LUT pour afficher l'image clown_lumi en niveaux de gris
r=0:1/255:1;
v=0:1/255:1;
b=0:1/255:1;
map=[r' v' b'];
image (I);
colormap(map)
% LUT si on souhaite compenser le gamma de l'écran
vect = 0:1/255:1;
gamma_r = 2.2;        % Niveau_sortie=Niveau_entrée^2.2 pour le rouge
gamma_v = 2.3;        % Niveau_sortie=Niveau_entrée^2.3 pour le vert
gamma_b = 2.1;        % Niveau_sortie=Niveau_entrée^2.1 pour le bleu
r = vect.^(1/gamma_r);
v = vect.^(1/gamma_v);
b = vect.^(1/gamma_b);
```

```
% on fabrique la LUT à l'aide des 3 vecteurs
map_gamma_inv = [r' v' b'];
% pour appliquer la LUT
figure
image(I)
colormap(map_gamma_inv);
```

Voici les résultats obtenus respectivement avant (image de gauche) et après (image de droite) l'utilisation d'une LUT pour compenser le gamma de l'écran:



ACTION : En vous inspirant des exemple précédents, synthétisez une LUT pour faire une inversion vidéo de l'image *CLOWN_LUMI*. Ce traitement consiste à réaliser le négatif (bien connu en photographie) de chaque plan de couleurs i.e. les niveaux 0 deviennent 255 et inversement, les niveaux 1 deviennent 254, ...

5 – Affichage des plans R-V-B

Chargez l'image couleur *CLOWN* dans votre répertoire de travail. Observez le type des données, et visualisez l'image. Visualisez plan par plan l'image couleur *CLOWN* en créant les LUTs adéquates pour les plans Rouge, Vert, et Bleu (on s'inspirera de l'exercice « *Éclatement d'une image couleur sous Matlab* » du chapitre 1).

Correction de l'exercice sur les LUT

1 – Ouvrez l'éditeur de commande par la méthode de votre choix (cf. exercice de prise en main Matlab, Chapitre 1).

2 – Dans votre répertoire de travail exécutez la commande:

```
I = imread('CLOWN_LUMI.BMP');
```

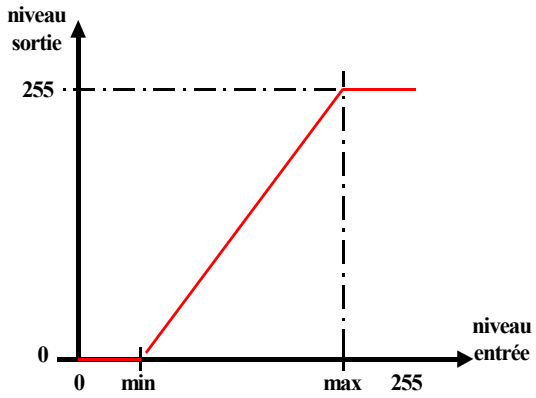
I est une matrice bidimensionnelle dans le cas d'une image en niveaux de gris. Le type de données de I est uint8 (unsigned integer).

3 – Il existe trois fonctions sous Matlab pour afficher des images : ***imshow***, ***image***, et ***imagesc***.

+ ***imshow*** affiche l'image contenue dans le fichier image ou la matrice correspondante. ***imshow*** appelle ***imread*** pour lire l'image depuis le fichier, mais les données de l'image ne seront pas stockées dans le workspace.

+ ***image*** affiche la matrice en argument comme une image. Elle n'accepte pas de fichier image en paramètre.
L'affichage repose sur les valeurs de la matrice ainsi que sur la carte des couleurs activées.

+ ***imagesc*** affiche la matrice en argument comme une image.
Contrairement à ***image***, elle effectue une remise à l'échelle des valeurs de la matrice avant l'affichage de manière à utiliser pleinement la carte des couleurs. Elle utilise donc la LUT suivante :



Afin de comparer visuellement ces trois fonctions, vous pouvez également taper la commande ***figure*** avant chacune des commandes ***image***, ***imagesc***, et ***imshow***. Ainsi Matlab ouvrira une nouvelle fenêtre d'affichage pour chaque nouvelle image.

```
% Visualisation des différences entre les fonctions d'affichage
figure
image(I) ;
figure
imagesc(I)
figure
imshow(I)
```

Exemple de trans-typage :

```
I = imread('CLOWN_LUMI.BMP');
J = double(I);
image(J);
```

4 – Voici un exemple de solution pour réaliser le négatif de l'image *CLOWN_LUMI* avec une LUT :

```
% Lecture des images
Im = imread('CLOWN_LUMI.BMP');
% Création de le LUT
r=1:-1/255:0;
v=1:-1/255:0;
b=1:-1/255:0;
lut=[r' v' b'];
image(Im)
colormap(lut)
```

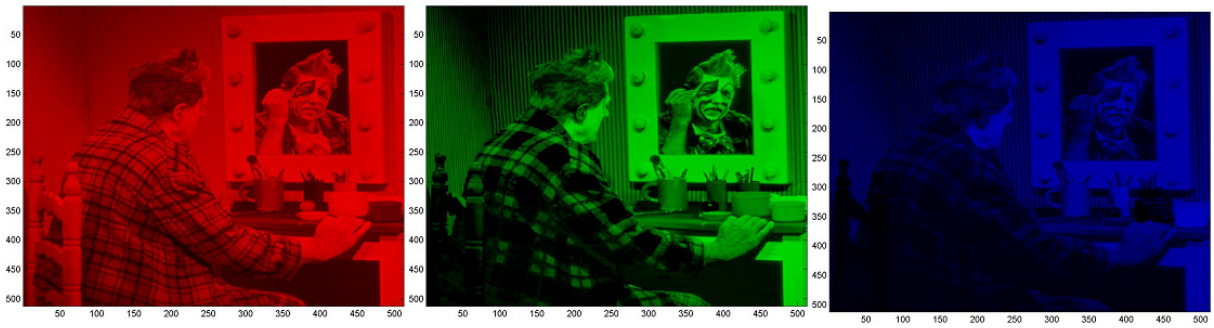
Voici les résultats obtenus :

Avec une LUT adéquate, il est donc possible d'afficher le négatif d'une image sans stocker aucune donnée supplémentaire de type « image » dans le workspace et sans modifier les données initiales de l'image.

5 – Dans le cas d'une image couleur la donnée est tri-dimensionnelle et de type uint8. Les trois plans d'une image couleur sont des plans monochromes. Pour les visualiser il faut créer des LUTs adéquates. Par exemple, pour visualiser le plan rouge, il faut créer une LUT dont les sorties ne soient que des niveaux de rouge.
Voici les commandes pour visualiser les 3 plans R-V-B de l'image couleur *CLOWN* :

```
Im=imread('CLOWN.BMP') ;
% Création des LUTs
vecteur=0:1/255:1;
r=vecteur;
v=vecteur*0;
b=vecteur*0;
lutr=[r' v' b'];
r=vecteur*0;
v=vecteur ;
lutv=[r' v' b'];
v=vecteur*0;
b=vecteur ;
lutb=[r' v' b'];
% Affichage des plans R-V-B
image(Im(:,:,1));       % Plan Rouge
colormap(lutr);
figure
image(Im(:,:,2));       % Plan Vert
colormap(lutv);
figure
image(Im( :, :,3));     % Plan Bleu
colormap(lutb);
```

Voici les résultats obtenus :



Remarque : Matlab est un outil de visualisation de données matricielles qui propose des LUT spécifiques. Ces LUT sont présentées à la rubrique « ***Supported Colormaps*** » de l'aide Matlab sur les ***colormap***. Notons cependant qu'il n'existe pas de LUT sous Matlab pour les images couleurs : il n'est pas possible de visualiser une image couleur en créant une LUT pour chacun des plans Rouge, Vert, Bleu.