Université Djilali Bounaama de Khemis Miliana **Faculté des Sciences et de la Technologie**
Département de Technologie/ Niveau et Spécialité: 1ère Année Master Telecom

Année universitaire : 2019/2020 –Semestre 2 **Matière :** Traitement d'images

TP N°01 : Toolbox de Matlab de Traitement d'images et de la vidéo

L'objectif de ce TP est de prendre en main les outils de traitement d'images les plus classiques à l'aide du logiciel Matlab. Matlab est un logiciel de calcul scientifique permettant de développer des solutions à des problèmes techniques. Il permet de réaliser du calcul numérique et de tracer des graphiques pour visualiser et analyser les données. Il dispose d'un langage et d'un environnement de programmation interactifs ainsi que d'outils pour concevoir des interfaces utilisateur graphiques. Matlab est associé `a des boites à outils appelé Toolbox permettant d'accéder à des fonctions spécifiques à un domaine d'application comme le traitement d'images par exemple.

Les TP de traitement d'images, réalisés avec Matlab nécessitent ainsi la toolbox Image Acquisition et la toolbox Image Processing. Les fonctions de cette dernière toolbox peuvent être listées en tapant la fonction **help images** dans l'´editeur de commande de Matlab. Pour obtenir un descriptif détaillé de ces fonctions, utiliser l'aide contextuelle en tapant la fonction help suivi du nom de la fonction. Avant de parcourir directement le sujet, vous pouvez découvrir les démonstrations associées au toolbox Image Processing en tapant la fonction **demos** dans le menu **help**.

Ce premier exercice est destiné à prendre en main les toolbox Image Processing et Image Acquisition.

Une image en noir et blanc est donc un tableau à 2 dimensions. Si *I* est la variable contenant les données d'une image à niveaux de gris, l'instruction I(1,1) renvoie la valeur du pixel de coordonnées (1,1). L'instruction I(:,1) renvoie les valeurs des pixels de la première colonne et l'instruction I(1,:) renvoie les valeurs des pixels de la première ligne...

1) Lecture, affichage et sauvegarde d'une image

Le chargement en mémoire d'une image se fait avec la fonction **imread**. Par exemple, la fonction suivante permet de lire une image et de placer son contenu dans une variable de type matrice :

I = imread('cameraman.tif');

Cette variable est alors visible dans le Workspace (espace des variables) de Matlab. La fonction **whos** permet d'afficher toutes les informations relatives aux variables en mémoire et la fonction **imfinfo** affiche les informations relatives à un fichier image.

L'affichage de l'image (ou de la variable) est réalisé par la fonction **imshow**. Ainsi les fonctions **figure; imshow(I);** ouvrent une nouvelle fenêtre pour y afficher l'image *I*.

a) Afficher l'image **« cameraman.tif »** et donner les caractéristiques de cette image. Etudier les options de la fonction **imshow.**

Les fonctions **imwrite** et **print** permettent la sauvegarde, respectivement, des images et des figures sous différents formats (**tif, jpg, bmp, png, gif, ……..**).

b) Enregistrer l'image **« cameraman.tif »** sous les formats suivants : JPEG, BMP, PNG, GIF. Ouvrir et afficher ensuite chacune de ces images, observer leurs différences et comparer les avec l'image d'origine. Mesurer ces différences avec l'image d'origine en calculant l'erreur quadratique moyenne.

c) Enregistrer l'image **« cameraman.tif »** au format JPEG avec différents niveaux de compression (mettre le paramètre **« Quality »** à 0, 25, 50, 75 et 100). Ouvrir et afficher ensuite chacune de ces images, observer leurs différences en taille et en qualité et comparer les avec l'image d'origine. Mesurer ces différences avec l'image d'origine en calculant l'erreur quadratique moyenne.

2) Types des images en mémoire : Matlab supporte 4 formats d'images :

• les images binaires, • les images d'intensités (à niveaux de gris),• les images couleur RGB, • les images couleur indexées. Il est possible de changer le format d'une image en utilisant les fonctions suivantes :

• **ind2gray:** indexé / intensité, • **ind2rgb :** indexé /RGB, • **rgb2ind :** RGB/ indexé,

• **rgb2gray :** RGB/intensité, • **im2bw:** intensité, indexé, RGB / binaire : c'est l'opérateur de binairisation.

Une image binaire peut être également obtenue en utilisant des opérateurs de comparaison et des opérateurs logiques. Par exemple, les instructions (I==seuil) ou ((I>=seuil bas) & (I<seuil haut)) permettent d'obtenir des images binaires par comparaison des niveaux des pixels d'une image I à des valeurs de seuils.

Le passage d'une image couleur à une image d'intensité correspond à la transformation des composantes R(rouge), G(vert),B(bleu) en la composante Y des systèmes de transmissions YIQ et YUV qui séparent l'information de luminance et de chrominance.

a) Ouvrir une image couleur de Matlab (peppers.png, fabric.png) et la convertir en une image d'intensité. Comparer cette image avec celle précédemment acquise directement en niveaux de gris en calculant l'erreur quadratique moyenne entre ces deux images.

b) Ouvrir l'image couleur acquise et enregistrée précédemment et extraire et afficher en niveaux de gris puis en couleur les images correspondant aux composantes R, G et B. Que peut-on observer ?

3) Palette des couleurs

Toute couleur peut être synthétisée à partir des 3 couleurs de base (R,G,B).Matlab contient une interface graphique permettant la visualisation de cette synthèse. Ouvrir le **help,** chercher **color palette** et exécuter **colorPalette GUI** pour obtenir une boite de dialogue qui vous permet de manipuler les différentes couleurs.

4) les séquences d'images et vidéo (multi frame array)

Une vidéo consiste en une série d'images en mouvement envoyées ave une certaine fréquence (nombre d'image /s).Pour créer une vidéo a partir d'une séquence d'image on utilise **immovie** et **implay** pour la voir.

Dans le « Help Navigator » taper « **multi frame array »** puis sélectionner **« Viewing Image Sequences »** pour voir quelques applications permettant la création d'une collection d'images retardées dans le temps comme trames de vidéo. Toutes les trames doivent avoir la même taille. Une trame peut être affichée par **« imshow »**

Les scripts matlab necessaries sont:

```
I = imread('cameraman.tif');
figure;  imshow(I);
imwrite(I,'cameraman.jpg');   %jpg, bmp,  png, gif, emf, eps
imwrite(I,'cameraman.bmp');

imwrite(I,'cameraman.gif');%imwrite(I,'cameraman.pcx');
imwrite(I,'cameraman.png');%imwrite(I,'cameraman.eps');
I_jpg=imread('cameraman.jpg');
I_bmp=imread('cameraman.bmp');
I_gif=imread('cameraman.gif');
I_png=imread('cameraman.png');
figure
subplot(221),imshow(I_jpg),title('image format jpg')
subplot(222),imshow(I_bmp),title('image format bmp')
subplot(223),imshow(I_gif),title('image format gif')
subplot(224),imshow(I_png),title('image format png')

%%=======chgt niveau de qualit? 0, 25, 50, 75 et 100
imwrite(I, 'cameraman1.jpg', 'Quality', 0);
imwrite(I, 'cameraman2.jpg', 'Quality', 25);
imwrite(I, 'cameraman3.jpg', 'Quality', 50);
imwrite(I, 'cameraman4.jpg', 'Quality', 75);
imwrite(I, 'cameraman5.jpg', 'Quality', 100);

I_jpg1=imread('cameraman1.jpg');
I_jpg2=imread('cameraman2.jpg');
I_jpg3=imread('cameraman3.jpg');
I_jpg4=imread('cameraman4.jpg');
I_jpg5=imread('cameraman4.jpg');

figure
subplot(231),imshow(I),title('image format tif')
subplot(232),imshow(I_jpg1),title('image format jpg_q0')
subplot(233),imshow(I_jpg2),title('image format jpg_q1')
subplot(234),imshow(I_jpg3),title('image format jpg_q2')
subplot(235),imshow(I_jpg4),title('image format jpg_q3')
subplot(236),imshow(I_jpg5),title('image format jpg_q4')
```

on refait cette procedure pour les autres images

```
%A = imread('fabric.png');

A = imread('peppers.png');
imshow(A)
 g_A = rgb2gray(A);

figure
subplot(221),imshow(A),title('image couleur originale')
subplot(222),imshow(g_A),title('image intensite')

R=A(:,:,1);G=A(:,:,2);B=A(:,:,3);

figure
subplot(221),imshow(A),title('image couleur originale')
subplot(222),imshow(R),title('image intensite Rouge')
subplot(223),imshow(G),title('image intensite vert')
subplot(224),imshow(B),title('image intensite bleu')

figure
subplot(221),imshow(A),title('image couleur originale')
```

```
subplot(222),imshow(g_A(:,:,1)),title('image intensite Rouge')
subplot(223),imshow(g_A(:,:,2)),title('image intensite vert')
subplot(224),imshow(g_A(:,:,3)),title('image intensite bleu')
% moon_tiff = imread('moon.tif');
% imwrite(moon_tiff,'moon.jpg');

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];
imshow(RGB,'notruesize');

%%%===programme
clear all
RGB1 = imread('peppers.png');

R=RGB1(:,:,1);G=RGB1(:,:,2);B=RGB1(:,:,3);

figure
subplot(221),imshow(RGB1),title('image couleur')
subplot(222),imshow(R),title('canal rouge')
subplot(223),imshow(G),title('canal vert')
subplot(224),imshow(B),title('canal bleu')

YIQ = rgb2ntsc(RGB1);
Y = YIQ(:,:,1);
I = YIQ(:,:,2);
Q = YIQ(:,:,3);
figure
subplot(221),imshow(RGB1),title('image couleur')
subplot(222),imshow(Y),title('canal Y')
subplot(223),imshow(2*abs(I)),title('canal 2*abs(I)')
subplot(224),imshow(2*abs(Q)),title('canal 2*abs(Q)')

HSV = rgb2hsv(RGB1);

H=HSV(:,:,1);
S=HSV(:,:,2);
V=HSV(:,:,3);
subplot(2,2,1), imshow(H)
subplot(2,2,2), imshow(S)
subplot(2,2,3), imshow(V)
subplot(2,2,4), imshow(RGB1)
%For closer inspection of the HSV color space, the next block of code displays
%the separate color planes (hue, saturation, and value) of an HSV image.

RGB2=reshape(ones(64,1)*reshape(jet(64),1,192),[64,64,3]);
HSV=rgb2hsv(RGB2);
H=HSV(:,:,1);
S=HSV(:,:,2);
V=HSV(:,:,3);
subplot(2,2,1), imshow(H)
subplot(2,2,2), imshow(S)
subplot(2,2,3), imshow(V)
subplot(2,2,4), imshow(RGB2)
```

rediger un compte-rendu résumant le travail effectué en commentant bien les resultants obtenus

TP N°02 : Traitement numérique des images par Matlab

I- Les bases d'affichage d'une image

Une autre fonction de base pour afficher une image est **« image »** dont le code est le suivant :

c=imread('cameraman.tif'); image(c),size(unique(c))
La commande complète pour afficher l'image est :
image(c),truesize, axis off, colormap(gray(247))
On peut ajuster la table des couleurs et voir son effet sur l'image :

1) *image(c),truesize,axis off, colormap(gray(512))*
2) *image(c),truesize,axis off, colormap(gray(128))*
3) *load mandrill,figure('color','k'),image(X),colormap(map)*
   axis off % Remove axis ticks and numbers
   axis image % Set aspect ratio to obtain square pixels

Changement du contraste :
c=imread('cameraman.tif');
cd=double(c);imshow(c)

figure,imshow(cd)
figure,imshow(cd/255)

figure,imshow(cd/512)

figure,imshow(cd/128)

Image binaire = image ayant deux valeurs 0 ou 1:**« *cl=c>120;imshow(cl)* »**

Les valeurs des images lues sous Matlab sont entières, mais dans certaines circonstances, on a besoin de travailler sur des valeurs réelles. La transformation pour passer d'entier à réel utilise la fonction ***im2double***.

img=imread('cameraman.tif'); figure;imshow(img);imgdbl=im2double(img); figure, imshow(imgdbl);imgint=im2uint8(imgdbl);figure;imshow(imgint); imwrite(imgint,'test.jpg','jpeg'); whos

Résolution spatiale : c'est la densité des pixels dans l'image : la grande résolution spatiale signifie que la plupart des pixels sont utilisés pour afficher l'image. La mise en œuvre se fait avec ***imresize***

x=imread('cameraman.tif');
x1=imresize(imresize(x,1/4),4); figure,imshow(x1); x2=imresize(imresize(x,1/8),8); figure,imshow(x2);
x3=imresize(imresize(x,1/16),16);figure,imshow(x3); x4=imresize(imresize(x,1/32),32);figure,imshow(x4);

Travail demandé N°1:

1- Taper la commande : **help imdemos**

Ceci vous donne une liste des images TIFF de la boite « Image Processing Toolbox ». Citer 5 images de cette liste. Pour chacune faites ce qui suit :

(a) déterminer son type (binaire, en niveaux de gris, vrai couleur or couleur indexée),

(b) déterminer sa taille (en pixels)

(c) donner une brève description de l'image. (what it looks like; what it seems to be a picture of)

2- Prendre une image NG, (**cameraman.tif** ou **…..tif**). Utiliser la fonction **imwrite**, pour l'écrire en fichiers de type JPEG, PNG and BMP. Quelle est la taille de ces fichiers ?

3- Répéter la question précédente pour:

(a) une image binaire. (b) une image couleur indexée. (c) une image en vraie couleur

4- Appliquer la commande **im2uint8** à une image format **.tiff** de votre choix avec la syntaxe : **e2=im2uint8(e).** Visualiser la sortie. Déduire le rôle de la fonction **im2uint8** sur l'apparence et les éléments de la matrice d'image?

5- Que se passe-t-il si on applique « im2uint8 » à l'image « cameraman »?

6- Faites une réduction de la résolution spatiale des images suivantes:

a- **cameraman.tif** b- l'image NG **trees.tif** c- **pout.tif** d- **eight.tif**

Pour chaque cas noter le point où l'image devient non reconnaissable.

II- Transformations ponctuelles sur l'image:

1- Opérations arithmétiques:

```
b1=uint8(double(b)+128);
b1=imadd(b,128);
b2=imsubtract(b,128);
imshow(b1),figure,imshow(b2)
b=imread('cameraman.tif'); whos b
b3=immultiply(b,0.5); %%or b3=imdivide(b,2) y=x/2 ou x*.5
b4=immultiply(b,2);%y=2x
```

```
%%b1=b+128; %erreur
b1=uint8(double(b)+128);
b1=imadd(b,128);
b2=imsubtract(b,128);
```

```
%y=x/2+128
b5=imadd(immultiply(b,0.5),12);
% or b5=imadd(imdivide(b,2),128);
bc=imcomplement(b);
imshow(bc)
```

2. Histogramme - seuillage

L'histogramme d'une image donne la répartition de ses niveaux de gris. Ainsi pour une image qui possède 256 niveaux de gris, l'histogramme représente le niveau de gris en fonction du nombre de pixels à ce niveau de gris dans l'image. L'histogramme donne donc une excellente idée de la séparation entre quelque chose qui est clair et quelque chose qui est foncé dans l'image.

```
img = imread('cameraman.tif'); histo = imhist(img,256); figure;plot(histo); p=imread('pout.tif');
imshow(p),figure,imhist(p),axis tight, ph=histeq(p); imshow(ph),figure,imhist(ph),axis tight
```

Typiquement, une utilisation de ce fait est le seuillage d'une image :

```
img=imread('saturn.tif'); figure;imshow(img); img=im2double(img);figure;

subplot(1,2,1);imshow(img); result=(img>0.5).*img; subplot(1,2,2);imshow(result);
```

2.1 Amélioration du contraste d'une image : Egalisation de l'histogramme

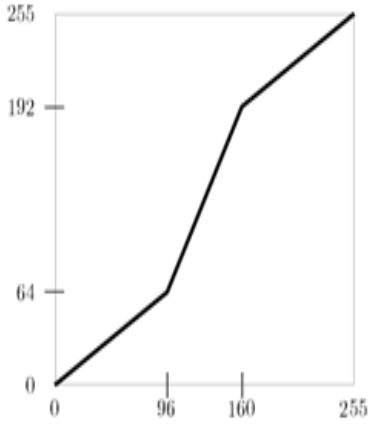
- `p=imread('pout.tif'); ph=histeq(p);imshow(ph),figure,imhist(ph),axis tight`
- `en=imread('pout.tif'); e=imdivide(en,4);imshow(e),figure,imhist(e),axis tight`
- `eh=histeq(e);imshow(eh),figure,imhist(eh),axis tight`

2.2. Table de conversion (Look Up Table :LUT): permet de faire des transformations ponctuelles sur l'image. Par exemple(1), LUT correspondant à une division par 2, du niveau de gris, est comme suit :

index	0	1	2	3	4	5	……	250	251	252	253	254	255
LUT	0	0	1	1	2	2	……	125	125	126	126	127	127

Code matlab: T=uint8(floor(0:255)/2); b2=T(b);

Exemple 2 :



$$y = \frac{61}{96}x\,, \qquad y = \frac{192-64}{255-96}(x-96)+64\,, y = \frac{255-192}{255-160}(x-160)+192$$

Ces équations peuvent être écrites comme suit :

$$y = 0.6667x\,,\ y = 2x-128\,, y = 0.6632x+85{,}8947$$

On peut construire cette table de conversion en tapant :

t1=0.6667[0:64]; t2=2*[65:160]-128; t3=0.6632*[161:255]+85.8947;*

T=uint8(floor([t1 t2 t3])); a2=T(b); imshow(b),figure,imshow(a2)

Travail demandé N^{0}2:

A) Image Arithmétique

1. Décrire LUT pour: (a) multiplication par 2, (b) complément d'image

2. Appliquer sur l'image b, la commande*: b2=imdivide(b,64); bb2=immultiply(b2,64);imshow(bb2)*

Commenter le résultat obtenu. Pourquoi le résultat n'est pas équivalent à l'image originale?

3. Remplacer la valeur 64 de la question précédente avec 32, puis 16.

B) Histogramme

1- La table suivante donne le nombre de pixels pour les niveaux de gris de 0……7 :

NG_i	0	1	2	3	4	5	6	7
n_i	3244	3899	4559	2573	1428	530	101	50

Tracer l'histogramme correspondant, puis faites une égalisation d'histogramme et tracer le résultat obtenu.

2- Refaites la question précédente pour les deux tables suivantes

a)

NG_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_i	20	40	60	75	80	75	65	55	50	45	40	35	30	25	20	30

b)

NG_i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_i	0	0	40	80	45	110	70	0	0	0	0	0	0	0	0	15

3- On donne une petite image comme suit :

Calculer l'histogramme et la table qui permet l'égalisation de cet histogramme.

12	6	5	13	14	14	16	15
11	10	8	5	8	11	14	14
9	8	3	4	7	12	18	19
10	7	4	2	10	12	13	17
16	9	13	13	16	19	19	17
12	10	14	15	18	18	16	14
11	8	10	12	14	13	14	15
8	6	3	7	9	11	12	12

4- Créer une image sombre avec :
c=imread('cameraman.tif'); [x,map]=gray2ind(c);

La visualisation de la matrice x, donne une version très sombre de l'image **cameraman**.

Appliquer une égalisation d'histogramme à cette dernière, et comparer le résultat avec l'image originale.

5- En utilisant **p** et **ph** de la section II-2, entrer la commande :

figure, plot(p,ph,'.'),grid on; Que voyez- vous ?

6- faire des essais avec d'autres images en niveau de gris.

III. Opérations géométriques

Les opérations géométriques classiques permises avec la boîte à outils de traitement d'images: rotation, changement de taille, découpage...

img=imread('cameraman.tif');figure;imshow(img);imgrot1=imrotate(img,3,'bilinear');
figure;imshow(imgrot1);imgrot2=imrotate(img,3,'bilinear','crop');figure;imshow(imgrot2)

Dans le premier cas, **imgrot1** est plus grande que **img**. Dans le second cas, le paramètre **'crop'** impose un découpage de l'image et la taille de l'image **imgrot2** est la même que celle de l'image **img.**

Le « **zoom »** permet d'agrandir une partie de l'image à l'aide de la souris, pour cela, il faut entrer la commande dans l'éditeur de Matlab puis sélectionner une zone de l'image à agrandir. Le bouton droit de la souris permet de revenir à la taille normale. **» zoom**

Un autre outil géométrique très utile pour détecter les niveaux de gris dans une image est la fonction de **« profil ».** On exécute cette fonction, on choisit à l'aide de la souris une ligne de l'image et on obtient le profil du niveau de gris le long de cette ligne.

img=imread('cameraman.tif'); figure;imshow(img); improfile