TP2 : binarisation et seuillage multiple d'images en niveau de gris

Dans ce TP, vous allez travailler sur des **images fixes** qu'il faut télécharger ici : http://ufrsciencestech.u-bourgogne.fr/~roudet/enseignement/TI/images.zip (images de tailles et de formats différents, en couleur ou NG, éventuellement bruitées ou à faible contraste).

Pour chaque exercice, il faudra créer une **fonction Matlab** permettant de réaliser le traitement demandé et vérifier son bon fonctionnement en testant le résultat sur plusieurs des images fournies.

I) Indications :

1) Voici un exemple de fonction Matlab pour réaliser l'inversion vidéo d'une image (négatif) :

```
function res = inversion (I)  % fonction qui prend en paramètre une image I
                              % et retourne une image res (négatif)
[m, n, can] = size(I);  % m=nb lignes, n=nb colonnes, can=nb canaux
res = zeros (m, n);     % image résultante (de même taille que I) :
                        % initialisée à 0 partout
if(can > 1)
    I = rgb2gray(I);  % si l'image est en couleur, la transformer en NG
end
res = 255 - I;      % l'inversion
figure
%divise la fenêtre en 1 ligne et 2 colonnes : pour afficher 2 images l'une
%à côté de l'autre
subplot(1, 2, 1)  %sélectionne le premier cadran de la fenêtre
colormap(gray(256))
imagesc(I);      %affichage de l'image originale
subplot(1, 2, 2) %sélectionne le deuxième cadran de la fenêtre
colormap(gray(256))
imagesc(res);   %affichage du résultat
title(strcat(['Image de taille ', num2str(m), 'x', num2str(n)]));
```

2) Utilisation de la **fonction inversion** définie précédemment (en supposant qu'on dispose de l'image « barbara512.png ») :

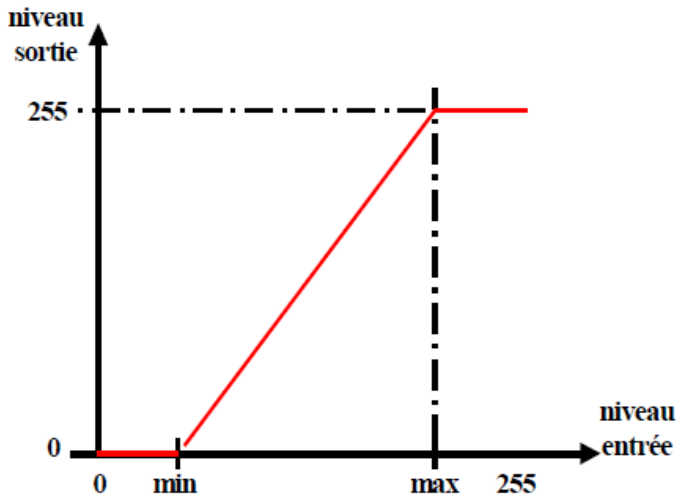
```
> Im = imread('barbara512.png');
> inversion(Im);
```

Commentaires : Im est une matrice bidimensionnelle dans le cas d'une image en niveaux de gris. Le type de données de Im est **uint8** (unsigned integer sur 8 bits).

3) Affichage d'une image :

L'affichage d'une image peut se faire avec les fonctions : **image, imagesc et imshow**.

- **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 (n'accepte pas de fichier image en paramètre). L'affichage repose sur les valeurs de la matrice qui sont :
  - associées aux index de **la carte des couleurs activée** (colormap)
  - ou directement interprétées comme **valeurs RGB** (true color)
- **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 (correspondant à un étirement d'histogramme) :



Comparer visuellement ces trois fonctions, en se servant éventuellement de la commande **figure** avant chacune des commandes **image, imagesc, et imshow**. Ainsi Matlab ouvrira une nouvelle fenêtre d'affichage pour chaque nouvelle image.

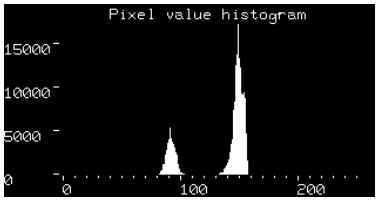
Par défaut la **carte de couleur** (colormap) utilisée par Matlab est la carte appelée **'jet'**, qui utilise un dégradé du bleu au rouge. Tapez *'help colormap' ou allez voir directement dans l'aide Matlab,* pour en savoir plus sur les cartes de couleurs disponibles.

II) Binarisation d'images

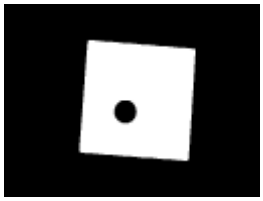
Créer la **fonction « Binarisation »** pour obtenir, à partir d'une image en niveaux de gris, **une image binaire.**

1) Le **seuil** aura été choisi en ayant **visualisé l'histogramme** au préalable (fonction **`imhist`** de Matlab). On pourra pour cela demander à l'utilisateur de **rentrer le seuil au clavier** (fonction **`input`** de Matlab).

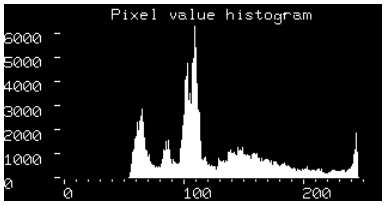
Exemple 1 : à retrouver en plus grand en Annexe 2

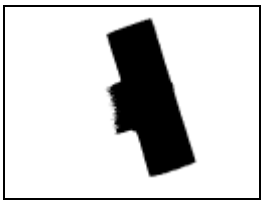


Résultat :



Exemple 2 : à retrouver en plus grand en Annexe 2





2) Le **seuil** sera déterminé **automatiquement** par la fonction **`graythresh`** (méthode d'**Otsu**).

Remarque : on pourra afficher dans la fenêtre de visualisation la **valeur du seuil** automatique.

Visualisez maintenant l'histogramme de l'image binarisée, que remarquez-vous ?

III) Seuillage multiple d'images

Créer la **fonction « Seuillage »** pour obtenir, à partir d'une image en niveaux de gris à **256 niveaux**, **une image quantifiée** sur **10 niveaux maximum** (10 classes)**.**

Le **nombre de niveaux** de l'image de sortie sera déterminé **après visualisation de l'histogramme** de l'image d'entrée. On précisera dans l'algorithme à quels niveaux on change de classe (seuils).

Annexe 1 : Images pour les tests

Utiliser le code Matlab suivant pour l'acquisition de ces images avec la webcam :
http://ufrsciencestech.u-bourgogne.fr/~roudet/enseignement/TI/acquis_test_M1.m