Traitement d'images TP1 : Bases

Objectif : Découvrir quelques méthodes de base du traitement d'images. Pour en savoir plus sur telle ou telle fonction de traitement d'images, vous pouvez toujours vous aider du **help** de Matlab. Il est conseillé de regarder la documentation offerte par le help pour toutes les fonctions que vous allez utiliser.

Quelques Généralités sur Matlab

Matlab est un logiciel de calcul scientifique permettant de développer des solutions à des problèmes techniques. Il utilise les données sous forme matricielle. Ce format est très compatible avec le format de représentation des images.

La fenêtre Matlab est décomposée en plusieurs onglets :

- Un onglet d'édition de commandes (Command Window)
- Un onglet de visualisation de l'espace des variables (Workspace)
- Un onglet de visualisation des fichiers du répertoire du travail (Current Folder)
- Un onglet permettant de visualiser le contenu des variables (Variables)
- Un onglet permettant d'éditer le contenu des fichiers (Editor)

Scripts et fonctions

Il est possible d'enregistrer une séquence d'instructions dans un fichier (appelé un M-file) et de les faire exécuter par MATLAB. Un tel fichier doit obligatoirement avoir une extension de la forme .m (d'où le nom M-file) pour être considéré par MATLAB comme un fichier d'instructions. On distingue 2 types de M-file, les fichiers de scripts et les fichiers de fonctions. Un script est un ensemble d'instructions MATLAB qui joue le rôle de programme principal. Si le script est écrit dans le fichier de nom nom.m on l'exécute dans la fenêtre MATLAB en tapant nom.

1 Lecture et visualisation :

a) pout.tif est une image fournie dans matlab. Pour charger cette image, utilisez la fonction *imread* :

```
I=imread('pout.tif') ;
```

La variable I est une matrice M x N. Vous pouvez obtenir ses dimensions en utilisant la fonction *size :*

```
[M, N] = size(I) ;
```

Pour visualiser les images, Matlab propose 2 commandes : *imshow* et *imagesc*. Voyons la différence : générer une image contenant des valeurs aléatoires entre 0 et 1000 :

```
I3 = rand(256).*1000;
```

Afficher l'image avec imshow puis essayez avec imagesc en executant les commandes suivantes :

```
Imagesc (I3) ;
axis image;
 axis off;
colormap(gray)
colorbar;
```

En effet imshow suppose que la donnée en entrée est une donnée normalisée (binaire ou affichant des intensités dans l'intervalle [0,255]). Par contre, imagesc accepte n'importe quelle donnée Matlab, la remet en échelle et la visualise en respectant le colormap par défaut ou celui spécifié. Pour que imshow visualise le même résultat, il suffit de lui spécifier l'échelle de l'image :
imshow(B,[0 1000]);

b) lire une image couleur et l'afficher en exécutant les commandes suivantes :

```
f = imread ( 'fleur.png' ) ; n = size ( f , 1 ) ;
figure ;
f1 = cat ( 3 , f ( : , : , 1 ) , zeros (n) , zeros (n) ) ;
f2 = cat ( 3 , zeros (n) , f ( : , : , 2 ) , zeros (n) ) ;
f3 = cat ( 3 , zeros (n) , zeros (n) , f ( : , : , 3 ) ) ;
subplot ( 2 , 2 , 1 ) ; imshow( f ) ; title ( 'image fleur' ) ;
subplot ( 2 , 2 , 2 ) ; imshow( f1 ) ; title ( 'composante rouge' ) ;

subplot ( 2 , 2 , 3 ) ; imshow( f2 ) ; title ( 'composante verte' ) ;

subplot ( 2 , 2 , 4 ) ; imshow( f3 ) ; title ( 'composante bleue' ) ;
```

Que fait la commande subplot ?

2. Exploration de l'image

a) Que valent les valeurs des trois composantes pour le pixel (245, 50) ? On peut obtenir cette valeur directement par inspection du tableau **f**, ou bien sur la figure avec l'outil **datacursor**. Tester les deux méthodes.

b) Tracez des coupes de l'intensité de l'image avec **improfile**. Que peut-on dire (qualitativement) quant à la régularité de l'image ?

c) Créez une version en niveau de gris, et la sauvegarder dans un fichier (utiliser **rgb2gray** et **imwrite**).

3. Manipulations d'histogramme

a) Ecrire une fonction qui permet de générer l'histogramme de l'image fournie en entrée.
b) Générer l'histogramme de l'image pout.tif en executant votre fonction.
c) Comparer votre résultat avec celui fourni par la commande hist ou histogram de matlab.

3.1 Contraste

a) On va modifier le contraste d'une image, défini comme l'écart type. Attention, lorsqu'on manipule les valeurs de niveau de gris, on doit convertir les valeurs **uint8** en **double**. On fera également attention à la conversion et au "scaling" lors de l'affichage. Exécutez les commandes suivantes, et expliquez la différence d'affichage pour les 2 dernières images :

```
f = double ( imread ( 'lena_gray.tif' ) ) ;
f2 = f + 0.5*( f-mean( f(:) ) ) ; % augmentation du contraste facteur 1.5
figure ( ) ; imshow( uint8 ( f ) ) ;
figure ( ) ; imshow( uint8 ( f2 ) ) ;
figure ( ) ; imshow( f2 , [ ] ) ;
```

b) Chargez et visualisez les 2 images *IRM1.jpg* et *IRM2.jpg* correspondant à une coupe IRM d'un patient atteint de sclérose en plaque, les 2 IRMs ayant été effectuées à 3 mois d'intervalle. Pour faciliter l'analyse de ces images et de l'évolution sous-jacente il peut être intéressant de visualiser la valeur absolue de la différence des 2 images. Essayez. Quel problème rencontre-t-on ? Proposez une solution pour améliorer le résultat.

3.2 Égalisation d'histogramme

a) Implémentez la fonction g et tester la, avec l'image pout.tif
g = egalise(f) retourne l'image g, version égalisée de f.

b) On s'intéresse ici au traitement par histogramme d'images couleurs. Peut-on généraliser l'égalisation d'histogramme au cas d'une image couleur ?
Proposez une manière de contourner le problème. Testez cette technique sur l'image mandrill.bmp.