TP 5 : Traitement d'images

I. Images à niveaux de gris

Dans cette partie, on utilisera des images prédéfinies sur Matlab : « cameraman.tif », « coins.png » et « rice.png ».

1. Lecture et affichage d'images

Pour lire et afficher une image, il suffit d'utiliser respectivement les commandes imread et imshow.

On se propose d'ouvrir l'image cameraman.tif.

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

imshow(I);
```

2. Dimensions d'images

Pour afficher les dimensions (nombre de pixels lignes et nombre de pixel colonnes) d'une image I, il suffit d'utiliser la commande size(I).

Si les dimensions seront utilisés ultérieurement, il faudra plutôt utiliser la forme [l,c] = size(I) qui permet de mémoriser le nombre de pixels lignes dans une variable l et le nombre de pixels colonnes dans une variable c.

Déterminer la taille de l'image cameraman.tif.

3. Inversion d'images

L'inversion de la dynamique d'une image à niveaux de gris consiste à remplacer la valeur de chaque pixel par son complément sur l'échelle des niveaux de gris. Ceci se résume par la ligne de commande : I1=255 – I ;

Appliquer cette commande sur l'image cameraman.tif. et comparer les deux images (l'originale et l'image inversée).

4. **Binarisation**

   Pour convertir une image à niveaux de gris en une image binaire, il faut choisir un seuil d'une valeur comprise entre 0 et 1 et utiliser la commande im2bw comme suit :

   ```
   I2 = im2bw(I, seuil) ;
   ```

   Pour l'image cameraman.tif, visualiser les résultats de binarisation correspondant à trois valeurs différentes du seuil (0.2, 0.5 et 0.8)/
   Interpréter l'effet de la valeur du seuil sur l'image binaire obtenue.

5. **Histogramme**

   La commande permettant d'afficher l(histogramme d'une image à niveaux de gris est imhist(I).

   Afficher les histogrammes de l'image cameraman.tif et son inverse et comparer les résultats obtenus.

6. **Morphologie mathématique**

   Pour pouvoir appliquer les opérateurs morphologiques (érosion, dilatation, ouverture et fermeture), il faut d'abord définir la forme et la taille de l'élément structurant en utilisant la commande strel.

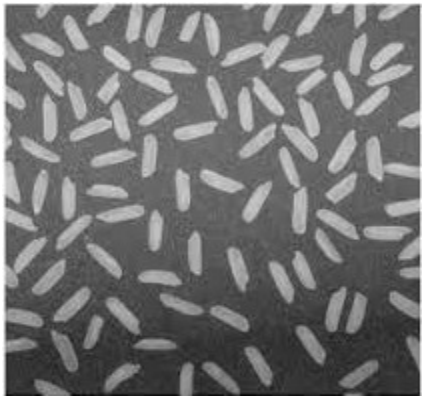
   Pour l'image étudiée, appliquer les opérateurs morphologiques pour différentes formes ('square', 'line'…) et tailles (3, 5, 9…) de l'élément structurant.

   ```
   se = strel('shape', size) ;
   I1 = imerode (I, se) ; % min(voisinage)
   I2 = imdilate (I, se) ; % max(voisinage)
   I3 = imopen (I, se) ; % dilatation(erosion)
   I4 = imclose (I, se) ;
   ```

7. **Algorithme des composantes connexes**

   Cet algorithme permet l'identification des différents éléments connexes contenus dans une image.

   Pour cette partie, utiliser une des deux images « coins.png » et « rice.png ».



```
Im = im2bw(I, seuil) ;
[L,num] = bwlabel(Im,4) ;
RGB = label2rgb(L) ;
figure, imshow(RGB)
```

Appliquer cet algorithme sur ces deux images et interpréter les résultats.

II. Images couleurs

Dans cette partie, on utilisera l'image « Fleurs.jpg » ou une image couleur de votre choix.

1. **Conversion d'une image RGB**
   Ecrire un programme Matlab permettant la conversion de cette image couleur en image à niveaux de gris, image ntsc et image hsv en utilisant respectivement les commande rgb2gray, rgb2ntsc et rgb2hsv.
2. **Morphologie mathématique**
   Appliquer les opérateurs morphologiques (érosion, dilatation, ouverture et fermeture) sur l'image couleur considérée pour plusieurs formes et différentes dimensions de l'élément structurant.
   Interpréter les résultats obtenus.
3. **Ajout d'un bruit synthétique**
   La commande imnoise permet d'ajouter un bruit à une image donnée.
   Prenons l'exemple d'un bruit de type « sel et poivre ».
   Ecrire un programme permettant de générer quatre images avec des quantités différentes de bruit synthétique de type sel et poivre comme suit :

IN1 = imnoise(I, 'salt & pepper', 0.2);
IN2 = imnoise(I, 'salt & pepper', 0.4);
IN3 = imnoise(I, 'salt & pepper', 0.6);
IN4 = imnoise(I, 'salt & pepper', 0.8);

Visualiser ces quatre images en plus de l'originale et déduire sur l'effet du bruit sur l'image.

4. Detection de contours

Plusieurs outils peuvent être exploités pour la détection de contours d'une image. L'un des outils les plus connus est l'analyse du gradient de l'image considérée qui correspond aux frontières entres des régions de caractéristiques différentes.

Appliquer l(algorithme suivant à votre image et afficher les résultats.

```
Rgb = imread('Fleurs.jpg');
I=rgb2gray(rgb);
Hy=fspecial('sobel');
Hx=hy';
Iy=imfilter(double(I),hy,'replicate');
Ix=imfilter(double(I),hx,'replicate');
Gradmag=sqrt(Ix.^2 + Iy.^2);
Figure, imshow(gradmag,[]);
title('Module du gradient');
```