Exercice Chapitre 2 – Binarisation

Cet exercice a pour objectif de vous présenter différentes réalisations de la binarisation d'une image sous Matlab. Chargez l'image *ISABE_LUMI.BMP* dans votre répertoire de travail et mettez à jour la liste des chemins dans le path browser.

1 – Lisez et affichez l'image (fonctions ***imread*** et ***image***). Affichez l'histogramme de l'image (fonction ***imhist***) afin de choisir un seuil de binarisation.

2 – La fonction ***im2bw*** (*image processing toolbox*) permet la binarisation d'une image. Consultez l'aide, créez et affichez l'image binaire de *ISABE_LUMI* à l'aide de la fonction ***im2bw***.

3 – On se propose maintenant de créer l'image binaire de *ISABE_LUMI* sans utiliser la fonction ***im2bw***. Créez un script de seuillage et de binarisation. Vous pouvez utiliser des instructions classiques (if, for, …), mais comme Matlab est un langage interprété, il est préférable de travailler en vectoriel (la commande ***find*** peut être utile ici).

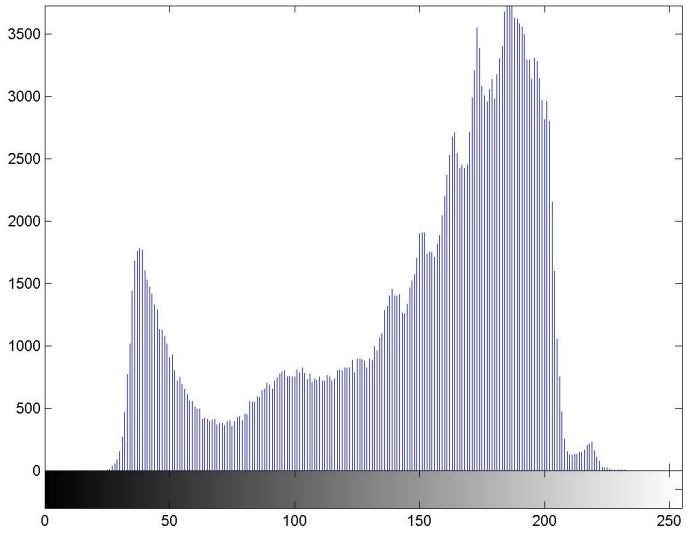
Correction de l'exercice : Binarisation

La binarisation est basée sur un seuillage brut. Cela signifie que si un pixel de l'image a une intensité supérieure à une certaine valeur de seuil, il lui sera attribué la couleur blanche sinon il sera noir. Ce procédé est réalisé sur chaque pixel de l'image. Nous obtenons donc une image comportant seulement deux niveaux (valeur 0 ou 1). L'image est binarisée.

1 – Après avoir rapatrié l'image ISABE_LUMI dans votre répertoire, entrez le script :

```
Im=imread('ISABE_LUMI.BMP');
r=0:1/255:1;
v=r;
b=r;
image(Im);
colormap([r' v' b']);
figure
imhist(Im);
```

On obtient alors l'histogramme suivant:

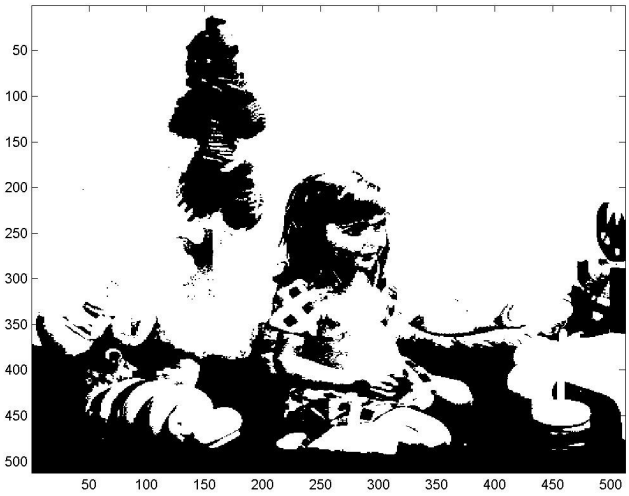


Le seuil de binarisation peut alors être choisi de différentes façons. Dans le cas de l'image *ISABE_LUMI*, on remarque que l'histogramme présente deux pics de population de pixels autour des niveaux de gris 75 et 190. On peut par exemple choisir la valeur de niveau de gris médiane entre ces deux pics. Néanmoins, il existe de nombreuses autres manières d'effectuer le seuillage : valeur moyenne de l'intensité sur l'ensemble de l'histogramme, valeur médiane de l'intervalle des niveau de gris [0, 255]), …

2 – On prend pour seuil de binarisation la valeur médiane 128 de l'intervalle des niveaux de gris [0, 255]. Cette valeur doit être normalisée sur l'intervalle [0, 1] pour être utilisée avec la fonction ***im2bw***. L'image binaire de *ISABE_LUMI* est alors obtenue par les commandes :

```
Im=imread('ISABE_LUMI.BMP');
ImBinaire=im2bw(Im,128/255) ;
imshow(ImBinaire);
```

On obtient alors l'image binaire suivante:



3 – On propose 3 autres scripts pour réaliser la binarisation (seuil=128).

a) Le premier script s'appuie sur l'utilisation de boucles « for ». Le but est de balayer l'ensemble des pixels de l'image. Un par un, on vérifie s'ils sont inférieurs ou supérieurs au seuil établi :

- si la valeur du pixel (i, j) de l'image avant binarisation est inférieure au seuil, le pixel (i, j) de l'image binaire sera noir (valeur 0) ;
- si la valeur est supérieure au seuil, le pixel (i, j) de l'image binaire sera blanc (valeur 1).

Voici le script Matlab pour réaliser ceci :

```
seuil = 128;
Im = imread ('ISABE_LUMI.BMP');
n= size(Im,1);    % Nombre de lignes de l'image
m= size(Im,2);    % Nombre de colonnes de l'image
      for i=1:n         % Balayage sur les lignes de l'image
            for j=1:m   % Balayage sur les colonnes de l'image
                        if Im(i,j) < seuil
                              ImBinaire(i,j) = 0;
                        else
                              ImBinaire (i,j) = 1;
                        end
            end
      end
imshow(ImBinaire);
```

b) Le second script utilise la fonction ***find*** de Matlab (évite les boucles imbriquées et permet un gain de temps de calcul). Cette fonction va chercher les coordonnées des pixels qui vérifient la condition passée en paramètre de la fonction ***find***.

```
Im=imread('ISABE_LUMI.BMP');
seuil = 128 ;
ImBinaire=Im;
ImBinaire(find(ImBinaire<seuil))=0;
ImBinaire(find(ImBinaire>=seuil))=1;
image(ImBinaire)
% Création d'une LUT pour afficher en noir et blanc
r=[0 1];
v=r;
b=r;
map=[r' v' b'];
colormap(map);
```

c) Le troisième script illustre également l'efficacité du calcul matriciel de Matlab (gain de temps de calcul par rapport aux boucles for) :

```
Im = imread ('ISABE_LUMI.BMP');
seuil = 128 ;
ImBinaire = Im > seuil ;
imshow(ImBinaire);
```

En entrant la commande « *ImBinaire* = *Im* > *seuil* », Matlab crée un masque logique *ImBinaire* de même taille que *Im*. Un pixel (i, j) du masque est à « 1 » quand la condition « Im(i, j)> *seuil* » est vraie et à « 0 » sinon. Le masque obtenu correspond donc à l'image binaire souhaitée.