TI TP5 : Transformations ponctuelles sous ImageJ

binôme : Benjamin Ruytoor et Aurore Allart
date : 20 février 2013

Introduction :

Dans ce TP, nous allons nous intéresser aux LUT : Look Up Table (table de correspondance). Ces tables nous permettent de faire correspondre une valeur de niveau de gris de l'image initiale à une valeur de niveau de gris transformé.

Pour chacune des questions suivantes nous allons utiliser l'image de Lena ci-dessous :

Figure 1 : l'image de Lena initiale.

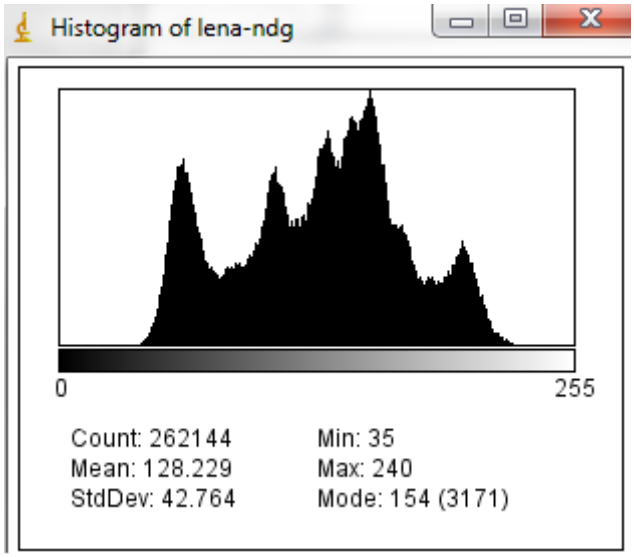


Figure 2 : histogramme de l'image ci-contre.

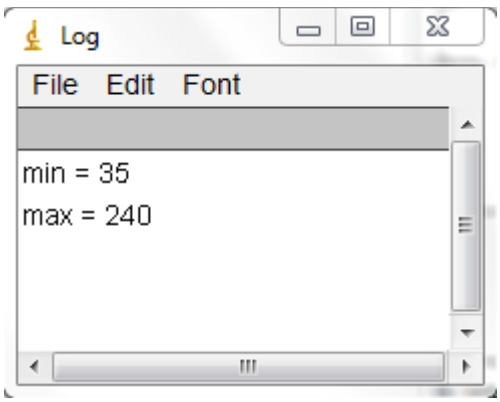
Pour une meilleure lisibilité des résultats, nous mettons dans ce rapport, l'image de Lena de départ et l'histogramme de niveaux de gris de celle_ci.

- ## Question 1

Voici la macro pour trouver le min et le max de niveaux de gris :

```
min = 255;
max = 0;

for (j=0; j<H; j++) {
        for (i=0; i<W; i++) {
                p = getPixel(i,j);
                if ( min > p){
                        min =p;
                }
                if ( max < p){
                        max =p;
                }
        }
  }
```



Voici le code de la loi de correction affine ainsi que le résultat obtenue :

```
//pour éviter les valeurs négatives et mettre le minimum à zéro
   for (i=0; i<min; i++)
   {
      reds[i] = 0;
      greens[i] = 0;
      blues[i] = 0;
   }

//pour étaler les valeurs des niveaux de gris
   for (i=min; i<max; i++)
   {
      reds[i] = ((i-min)*255/(max-min));
      greens[i] = ((i-min)*255/(max-min));
      blues[i] = ((i-min)*255/(max-min));
   }

//mettre le maximum à 255
   for (i=max; i<reds.length; i++)
   {
      reds[i] = 255;
      greens[i] = 255;
      blues[i] = 255;
   }
```

Figure 3 : l'image de Lena modifiée.

- ## Question 2

Comme on peut le constater sur l'histogramme, la modification des niveaux de gris de la question précédente, n'est effectif que sur l'image car les LUT ne modifie que le rendu visuel et non la valeur de chaque pixel.

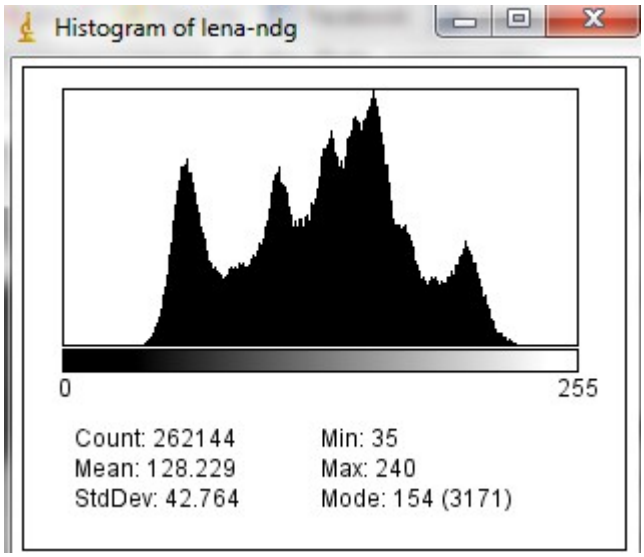


Figure 4 : histogramme de l'image précédente non modifié.

- **Question 3**

Voici le code permettant de modifier chaque pixel de l'image, ainsi que la validité par un histogramme.

```
for (j=0; j<H; j++)
        {
        for (i=0; i<W; i++)
                {
                p = getPixel(i,j);
                setPixel(i,j,reds[p]);
                }
        }
```

Comme on peut le constater sur l'histogramme, chaque pixel de l'image a eu sa valeur modifier en fonction des calculs de la question 2.

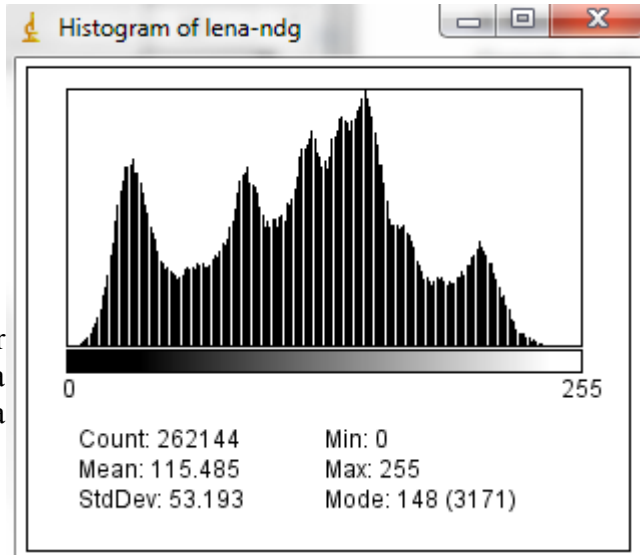


Figure 5 : histogramme modifié.

- **Question 4**

Voici la macro qui modifie les niveaux de gris des pixels par une égalisation d'histogramme et qui affiche le résultat.

```
// récupération de la taille du plan de fourier

W = getWidth();
H = getHeight();

// déclaration des LUTs
   t = newArray(256);

// Récupération des LUTS
getHistogram(values, counts, 256);
for (i=0; i<255;i++){
        res=0;
        for(j=0;j<i;j++){
                res=res+counts[j];
        }
        t[i]=(255/(H*W))*res;
}

for (j=0; j<H; j++) {
        for (i=0; i<W; i++) {
                p = getPixel(i,j);
                setPixel(i,j,t[p]);
        }
 }
```

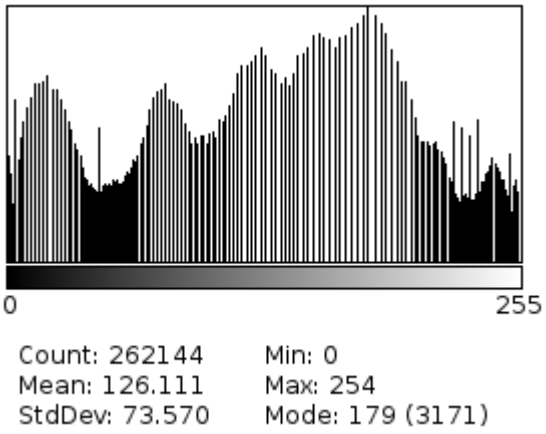


Figure 6 : histogramme égalisé.

Figure 7 : image de Lena égalisée.

- ## Question 5

On peut voir l'égalisation comme un moyen d'élargir le contraste de l'image, en aillant le plus de niveaux de gris possibles. Mais cela entraîne des trous blancs, c'est à dire des niveaux de gris jamais utilisés (voir la figure 6 et 7).
La sous-quantification réduit le nombre de niveaux de gris. Par conséquent, cela diminuera la gamme de l'histogramme et donc supprimera ces trous.

On peut donc dire qu'en réduisant le nombre de niveaux de gris, on comprime la répartition des niveaux de gris après égalisation.

L'avantage de cette sous-quantification après égalisation, c'est qu'elle permet de réduit la taille (en ko) de l'image, tout en gardant un maximum de détails de celle-ci.