LACOUTURE Amélie
VITRY Florent
BONARD Céline

L'IMAGERIE

Etudier une image à partir du logiciel scilab a été notre objectif. Mais pourquoi « l'imagerie » ? Sans doute car parmi les autres sujets proposés c'est celui qui nous a interpellé tous les trois, et qui paraissait le plus concret et intéressant pour nous. Pour notre étude de l'imagerie, nous avons du commencer par télécharger un autre logiciel : SIP (« scilab image processing ») qui nous a permis d'utiliser des fonctions de l'image. Tout d'abord nous verrons comment afficher et lire une image sur scilab, puis nous avons étudierons le contraste des images, et enfin nous vous présenterons les autres recherches pour modifier une image.

Après s'être placé dans le même répertoire que le logiciel SIP, que nous avons téléchargé pour étudier l'imagerie, voici l'ensemble de nos recherches :

LIRE ET AFFICHER UNE IMAGE AVEC SCILAB

Pour lire une image, il existe une fonction appelée *imread* permettant de transformer une image en un tableau de valeurs correspondantes aux pixels de chaque point de l'image.

Voici la manière dont elle s'utilise à partir d'une image (appelée « image.png ») enregistrée sur le répertoire où nous nous trouvons :

```
Tableau = imread ('image.png')
```

→ Ce qui nous rend le tableau de valeurs correspondant à notre image.

Pour afficher une image, il existe une fonction appelée *imshow* permettant, à partir d'un tableau, d'afficher notre image sur scilab.

Voici la manière pour afficher une image à partir d'un tableau de valeurs comprises entre 0 (couleur désignant le blanc) et 255 (couleur désignant le noir) : prenons l'exemple précédent (on a alors créé le tableau de valeurs de notre image initiale)

```
Image = imshow (Tableau)
```

→ Ce qui nous renvoie notre image « image.png »

C'est donc en appliquant successivement imread à une image, puis imshow au tableau obtenu, que l'on peut reproduire notre image, identique à l'originale, mais sur scilab.

MODIFIER LE CONTRASTE D'UNE IMAGE

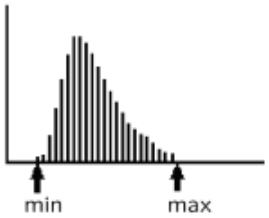
Avant tout, il nous a fallu créer une fonction permettant de tracer l'histogramme des fréquence des pixel dans l'image choisie, mais surtout de construire un tableau dont les valeurs représentent les fréquences des pixels de 0 à 255, c'est-à-dire chaque indice i correspond à un pixel (rangé dans l'ordre croissant) auquel est associé une valeur désignant le nombre de point de l'image ayant pour pixel i.

Voici l'écriture de cette fonction :

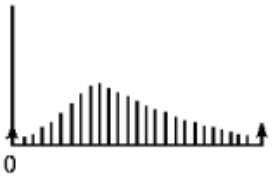
```
Function [tab] = histogramme (x)
 nb = prod (size (x))
 Data = matrix (x, 1, nb)              →  transforme notre tableau  pris en argument en matrice
 Tab = zeros (1, 256)                  → cré un tableau de zéros à un ligne et 256 colonnes
      For i = 0:255
           tab (i+1) = length (find (data==i)      →cette boucle for à permis à remplir
      end                                            Tab avec les différentes fréquences de pixel
histplot (256, sort (matrix (x, 1, nb)))      → construit l'histogramme des fréquences des pixels
endfunction
```

Cette fonction prendra en argument un tableau de valeurs comprises entre 0 et 255, soit, par exemple, x = imread ('image.png').

D'après l'histogramme obtenu par cette fonction, on peut remarquer qu'il existe un minimum min et un maximum max tel que pour tout i< min, la fréquence associé à ce i est nulle, de même pour tout i> max

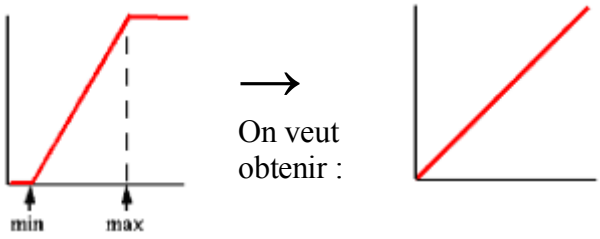


Le but de la fonction contraste que nous allons vous présenter par la suite est d'étendre l'histogramme de sorte que min soit en 0 et max en 255, c'est-à-dire de la façon suivante :



Ainsi le contraste de notre image sera amélioré.

Ce changement peut s'expliquer aussi à travers des fonctions :



Voici donc l'expression de notre fonction contraste, qui devra, après avoir appelé la fonction histogramme, trouver min et max, puis ensuite étendre l'histogramme de la manière illustrée précédemment.

```
Function [y] = contraste (x)
Tab=histogramme(x)                    → permet d'obtenir le tableau des fréquences des pixels
T= find (Tab==0)              →cré le tableau contenant les pixels pour lesquels la fréquence=0
For i = 1 : (length (T) -1)
      If  T(i+1)-T(i) <>1 then        → boucle qui permet de trouver le minimum
          Lmin = T(i)
          Break
      End
   End
   For j = length (T) : -1 : 2
      If  T(j) – T(j-1) <>1 then      → boucle qui permet de trouver le maximum
          Lmax = T(j)
          Break
      End
   End
y = min (max ( (256/ (Lmax – Lmin))* x – (256/ (Lmax – Lmin))*Lmin , 0) , 255)
imshow (y)                            →affiche l'image initiale mais avec un contraste amélioré
Endfunction
```

Cette fonction prend en argument un tableau, par exemple x=imread ('image.png').
Elle va donc permettre, à partir d'une image initiale, de modifier le contraste de cette image.

Voici un exemple d'utilisation de cette fonction contraste :

→ On applique la fonction contraste

Et enfin nous avons cherché à élargir nos recherches pour modifier une image.

<u>TRANSFORMATION BINAIRE</u>

Il s'agit de transformer une image à plusieurs couleurs en une image à seulement deux couleurs.

Pour cela, il existe la fonction *im2bw* prenant en argument un tableau de valeurs et un nombre compris entre 0 et 1 (0 représente la couleur blanche et 1 la couleur noire).

Ainsi on effectue les applications suivantes :

```
Tab = imread (image)
Bw = im2bw ( Tab, x)
imshow ( Bw, 2)
```

Où x est un nombre décimal compris entre 0 et 1.

On obtient alors une image à deux couleurs.

ROTATER UNE IMAGE

Il existe une fonction appelée *mogrify* qui permet d'effectuer une rotation de l'image. On l'utilise de la manière suivante :

```
Im2 = mogrify (x, ['rotate', 'y'])
Imshow (Im2)
```

Où x est un tableau, par exemple x=imread ('image.png'), et on fait une rotation de l'image d'angle y.

Ainsi on peut faire une rotation d'une image dans scilab.

Pour conclure, toute cette étude sur l'imagerie nous a permis de découvrir un logiciel, alors inconnu, certaines de ses fonctions, bien que ayons manqué de temps pour continuer nos recherches, et enfin cette étude nous a permis de créer nous-même des fonctions. Mais nous aurions pu élargir nos recherches si nous avions connu préalablement l'existence du logiciel SIP, et toutes les fonctions qu'il contient. Mais quoi qu'il en soit ce fut avec intérêt et curiosité que nous avons effectué ce travail.