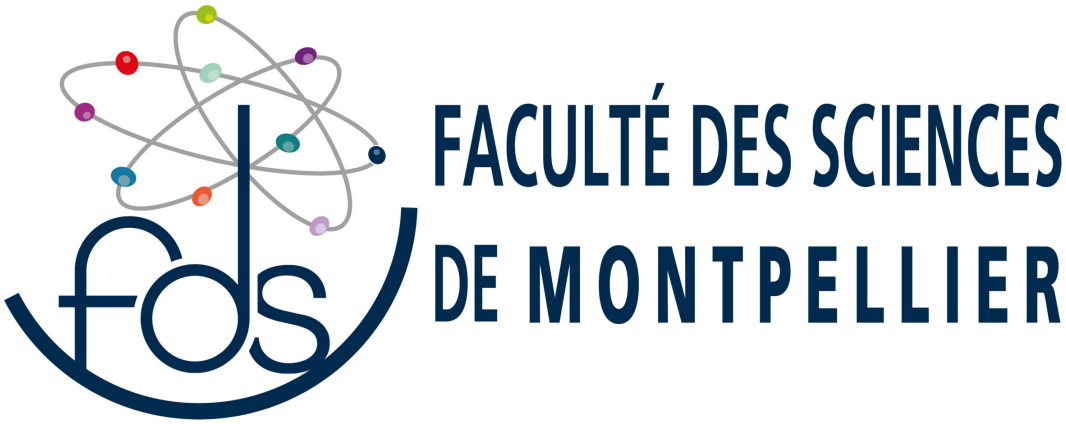
Université de Montpellier

M1 IMAGINE - Analyse et Traitement de l'image
Compte Rendu TP1 - Base du traitement d'image

Etudiant :
Guillaume Bataille

Encadrant :
Noura Faraj
Marc HARTLEY

Année 2022-2023



0.1 Contexte et objectif

Aujourd'hui, nous avons effectué un rappel des bases du traitement de l'image vu en L3 dans l'UE Données multimédia.

L'objectif de ce tp est donc de manipuler la librairie fournie par Mr William Puech afin d'effectuer des histogrammes, des profils de lignes/colonnes, des seuillages etc..
Afin de tester notre code, voici les deux images que nous choisissons de manipuler (proviennent de la librairie d'image ppm/pgm fournie par Mr William Puech).

FIGURE 1 – Image ppm (couleur) et pgm(niveau de gris) choisies pour ce tp

Sommaire

1. Seuillage (niveau de gris)

Pour pouvoir seuiller notre image stocké dans un vecteur, nous allons parcourir tout les pixels et tester la valeur en niveau de gris de ces derniers. S'il est inférieur au seuil, on le mets noir, sinon on le mets en blanc. Cela aura pour conséquence d'obtenir une image en noir et blanc.

FIGURE 1.1 – seuil = 220, trop peu de pixel blanc

FIGURE 1.2 – seuil = 90, trop eu de pixel noir

FIGURE 1.3 – seuil = 160, seuillage idéal en cherchant manuellement

2. Seuillage multiple S1, S2, et S3 (niveau de gris)

Maintenant, on essaye d'effectuer un seuillage multiple sur une image en niveau de gris. Le but est le même que le précédent, un parcours de tout les pixels mais les test sur le pixel sont en fonction de plusieurs seuil (dans la version qu'on implémente il y a le choix entre 1,2 ou 3 seuil). Les images seront donc en 0 et 255 (1 seuil), en 0, 128, 255 (2 seuils) ou en 0, 85, 170, 255 (3 seuils).
On voit que l'image gagne en niveau de détail et c'est logique car il y a plus de nuances de gris.

FIGURE 2.1 – Seuillage multiple S1 90 - S2 160 - S3 220

3. Profil d'une ligne ou colonne (niveau de gris)

Pour rappel, un profil ligne (ou colonne) d'une image consiste à lire et à récuperer la liste des intensités des pixels d'une ligne(ou colonne). C'est très pratique de pouvoir le faire afin de detecter des changement brusque d'intensité pouvant être interprété comme un contour.

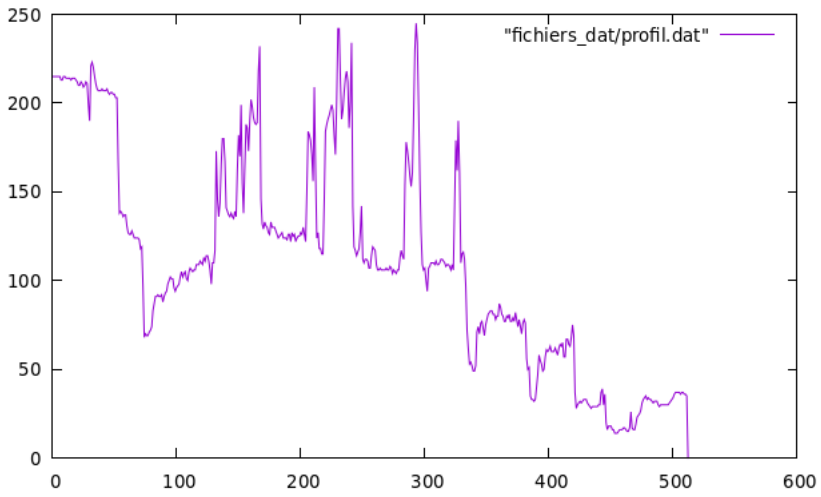


FIGURE 3.1 – profil de la ligne 250 de l'image originale (plus au moins au centre de l'image) via gnuplot

On peut voir plusieurs mod (pics). Cela se traduit par des changements d'intensité et donc des contours (surement l'écriture sur l'écriteau).

4. Histogramme (niveau de gris)

L'histogramme retranscrit les occurences des 256 niveau de gris sous forme graphique. On parcours donc tout les pixels et dès qu'on rencontre un niveau de gris [0,255], on ajoute une occurence de plus dans le tableau d'histogramme.

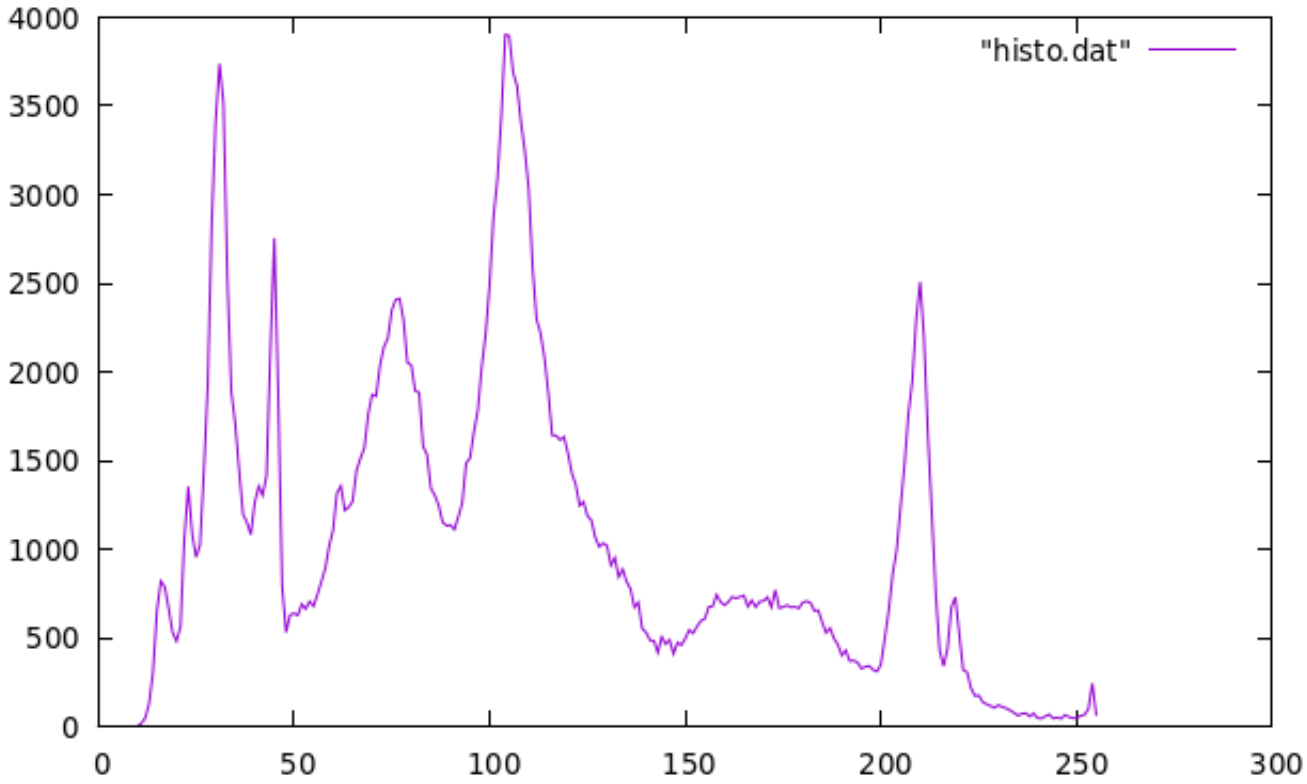


FIGURE 4.1 – histogramme de l'image originale via gnuplot

Plusieurs mod sont observés, et via lecture de l'histogramme on peut voir que l'image est a global plus sombre (plus de valeur au alentours de 0) que claire, chose confirmé après vérification sur l'image originale.

5. Histogramme RGB (couleur)

Maintenant qu'on a vu des manipulations d'images pgm (en niveau de gris), on s'attarde maintenant sur des images ppm (RGB).
En réalité, le travail est le même. Ici, pour l'histogramme on continue le parcours de tout les pixel mais dans le tableau de pixel on a : RGB RGB RGB etc..., on compte donc séparément la première, deuxième et troisième composante.

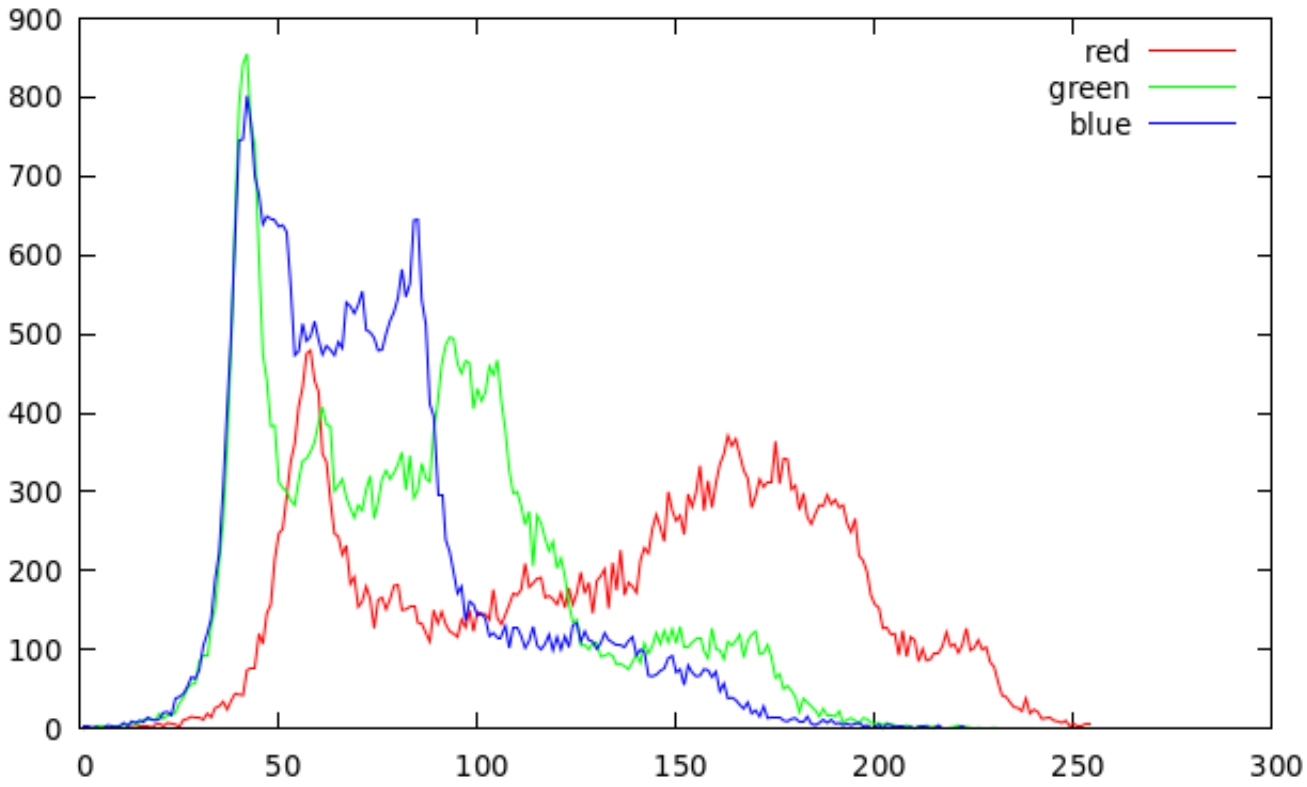


FIGURE 5.1 – Histogramme de lena RGB

6. Seuillage RGB (couleur)

Avec le code fourni (test_couleur.cpp) on a déja un seuillage effectif sur une image couleur.

FIGURE 6.1 – Seuillage de lena avec le même seuil (100) sur les 3 composantes

Cependant on seuille ici en passant sous silence l'existence de 3 composantes RGB.
On essaye donc d'appliquer un seuil sur chaque composante. Et une fois qu'on a saisi le concept, c'est répétitif. Ici on parcours chaque pixel par composantes (RGB) et on récupère 3 seuils (SR, SG, SB) qu'on utilise comme valeur de seuil sur chaque composantes.

FIGURE 6.2 – Respectivement des seuilsRGB 160-120-80 et l'autre des seuilsRGB 90-90-110

7. Seuillage automatique (niveau de gris)

En effet, lors de la premièrer question, nous avons determiner un seuil optimisé en testant pleins de seuil différents. Il serait interessant d'avoir un seuillage automatisé qui ne demande pas des tests et un jugement de l'utilisateur sur la pertinence de celui-ci.
On décide de partir sur un algorithme d'Otsu (vu en L3 - bonus de marc hartley) qui est un algorithme qui determine un seuil idéal via un histogramme. Le but est de tester les seuil et de conserver celui qui maximise les variances des deux groupes de pixel.

Pour rappel, la variance est la dispertion autour de la moyenne. Chercher donc le seuil [0,255] qui maximise cette variance est pertinent pour avoir deux groupes de pixel equitable.
Voici le code :

```
1   // PARTIE OTSU
2
3   //Somme des grey_level utilisé par l'algorithme d'otsu
4   int sum = 0;
5   int seuil;
6   for (int i = 0; i < maxGris; i++)
7   {
8     sum += i * Histo[i];
9   }
10
11  //Variable pour somme partielles
12  int sumB = 0;
13  int wB = 0;
14  int wF = 0;
15
16  int varMax = 0; // Variance Max
17  seuil = 0;      // Le seuil qui va être compute
18
19  //Parcours de chaque niveau de gris[0,255] pour trouver le meilleur seuil
20  for (int i = 0; i < maxGris; i++)
21  {
22    wB += Histo[i];
23
24    if (wB == 0) // Si y'a pas de pixel noir [0]
25    {
26      continue; // On skip (car division par 0 après)
27    }
28    wF = nTaille - wB;
29    if (wF == 0) // Si aucun pixel blanc [255]
30      break;     // On skip (car division par 0 après)
31
32    //Maj de la somme partielle
33    sumB += i * Histo[i];
```

```
34
35      //Moyenne des grey_lvl de chaque grp
36      int mB = sumB / wB;
37      int mF = (sum - sumB) / wF;
38
39      //Calcul de la variance entre les classes - formule internet
40      int var_mid = wB * wF * (mB - mF) * (mB - mF);
41
42      //Update si necessaire
43      if (var_mid > varMax)
44      {
45        varMax = var_mid;
46        seuil = i;
47      }
48    }
49    // FIN OTSU - seuil final automatique dans "seuil";
```

FIGURE 7.1 – Seuillage via algorithme d'Otsu - 128 pour la pancarte et 113 pour les loutres

8. Conclusion

Ce tp nous a appris a manipuler la librairie fournies par Mr William Puech afin de manipuler des images (ppm ou pgm) stockés dans des vecteurs/tableau d'int.
Cela nous a permis de traiter ces dernières via différentes méthodes, toutes ayant comme point commun le parcours des pixels suivi d'un traitement.