PROJET PROGRAMMATION

DEME Rémy
LAPORTE Nathan
PICHARD Thibault

PHASE 1

Pour réaliser notre projet, nous avons commencé par analyser la composition du format BMP. Nous avons déterminés qu'il fallait, pour pouvoir traiter une image de façon optimale, traiter dans un premier temps l'en-tête, puis le corps de l'image. Le premier défi a été de lire l'image et de la stocker sans modifier le fichier original. Nous avons donc décidé de créer une copie de l'image qui sera traitée.

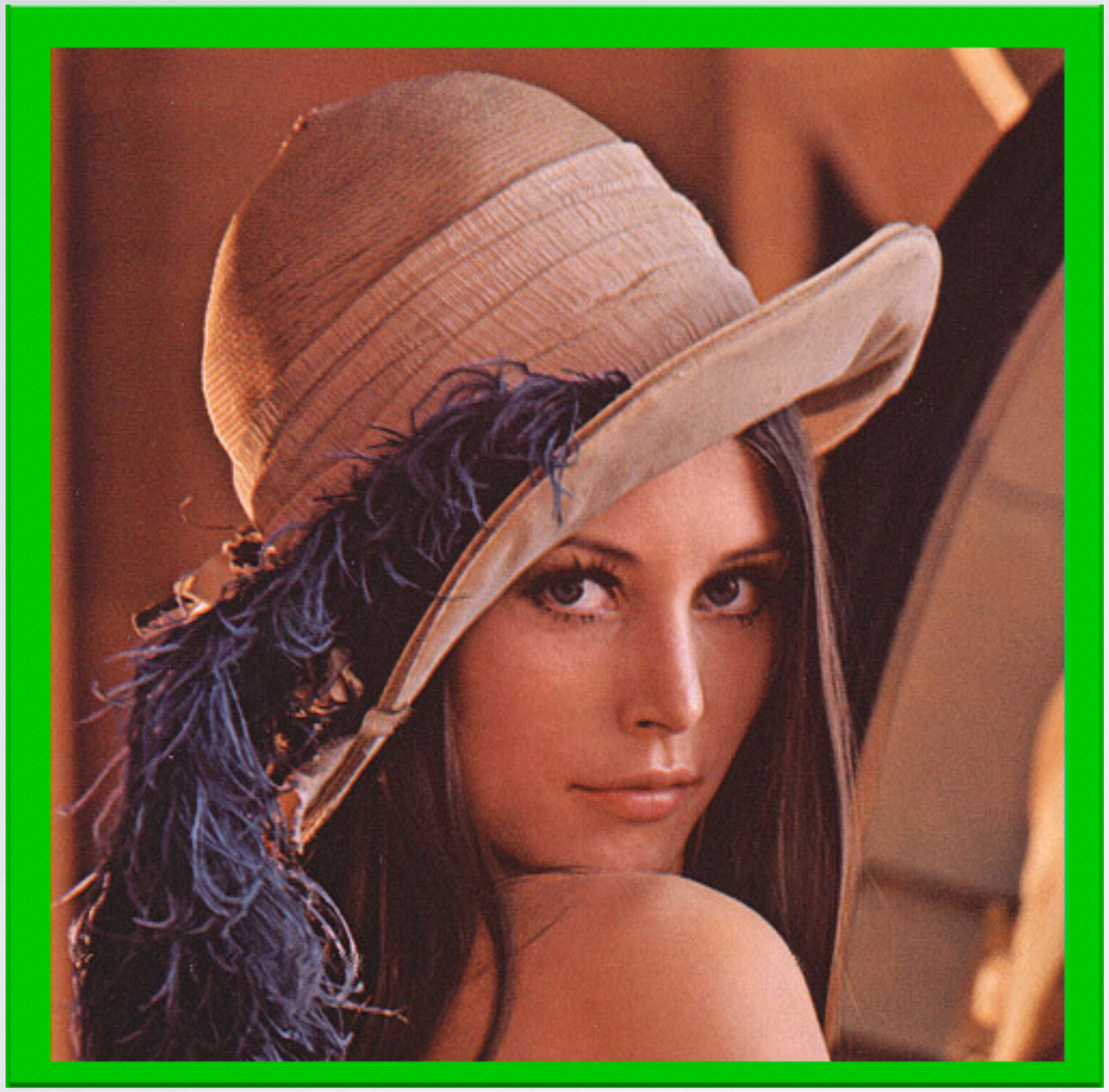
Ensuite, nous nous sommes décidés à traiter les deux cas (couleur ou N&B) différemment, afin de ne pas générer de conflit, le codage des pixels couleur se faisant sur 3 octets, alors que celui des pixels en nuance de gris se fait sur 1 octet.

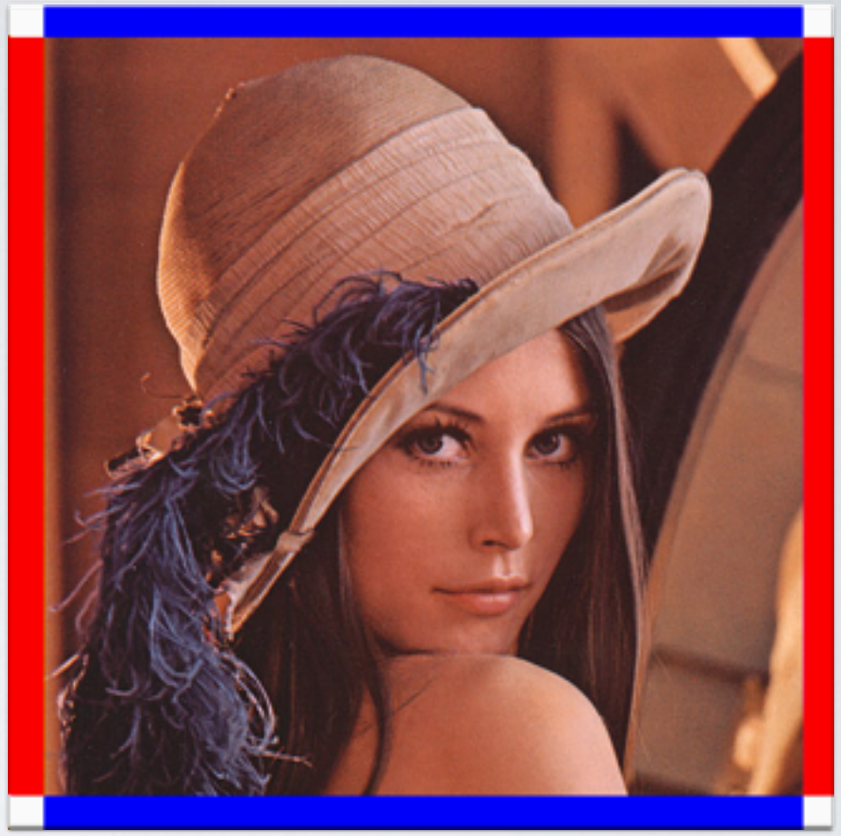
Par souci de simplifier le code, nous passons par chemin absolu et non par argument, ce qui permet en sus de lancer le programme sans passer par l'invite de commande

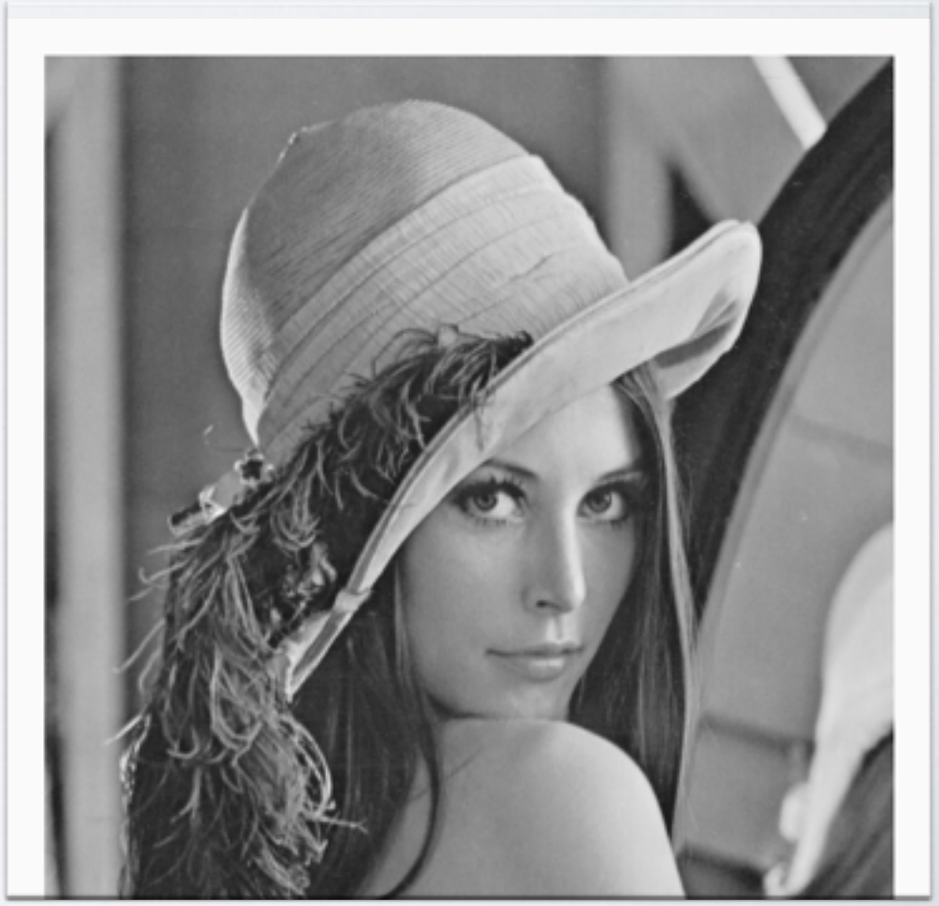
La partie la plus compliquée était de comprendre comment fonctionnait le header, d'autant plus qu'il existe des différences entre Windows et Linux (donc OS X), concernant le champ réservé, celui-ci étant prévu pour gérer la transparence. De plus, il y avait des problèmes d'alignement mémoire. En effet, lors du stockage dans la mémoire, le processeur met la variable à un emplacement multiple de sa taille. Ainsi, si l'on considère une structure comprenant un « char » et deux « int » (dans cet ordre), et que la mémoire commence à 0, le « char » est mis à l'adresse 0, et le premier int doit être mis à l'adresse 8, soit le plus proche multiple de 8 disponible (un « int » est stocké sur 8 octets en 64-bits). Entre les adresses 1 et 7, il y a 7 octets de paddings qui sont inutiles en soit. Or pour que le header de l'image soit lu et interprété correctement, il faut que tout soit aligné, ce qui est fait en utilisant la commande pragma pack.

Nous avons décidé de stocker les pixels comme types « unsigned char », car un pixel ne peut prendre une valeur négative, et qu'il doit avoir une valeur contenue entre 0 et 255, ce qui est la même chose que la table ASCII servant pour les « char ». On a ainsi une optimisation de la mémoire.

Pour le coloriage des bords, une fois le fichier copié, il n'est pas compliqué de se déplacer dans la matrice pour modifier les valeurs des couleurs, surtout si celles-ci sont fixes et ne dépendent pas des pixels adjacents.







★★ PHASE II

Le sujet choisi est celui de la segmentation par la méthode des K-Moyennes. Cette méthode à pour but de séparer les différents éléments d'une image en fonction de paramètres définis.

La méthode consiste à place des points (le plus souvent aléatoirement) puis d'assigner des éléments (dans notre cas les pixels) aux centres les plus proches. Une fois que tout cela est terminé, on replace des points, qui seront les barycentres (centres d'équilibre) des segments (ou groupes, ou encore clusters) précédemment traités. On refait ces étapes jusqu'à ce que plus aucun mouvement de centre ne soit à faire. L'image est alors considérée comme segmentée.

Il apparait plusieurs phénomènes :

- Selon les positions initiales des centres, plus ou moins d'opérations sont à prévoir. Dans certains cas, il peut arriver que l'on se trouve dans une boucle infinie, où à chaque fois que l'on replace les centres, on créé de nouveaux segments.
- Plus il y a de centres (donc de clusters), plus la segmentation est fine.

Il est apparu assez tôt que ce sujet était complexe mathématiquement, car il y a une part aléatoire, et une part de géométrie et de calcul (position des centres, barycentres...

Une que le clustering a fonctionné, nous avons choisi de représenter chaque cluster en fausse couleur, permettant de s'assurer que tout est à la bonne place.

Nous avons aussi ajouté d'autres fonctionnalités, comme un histogramme, ou des filtres.

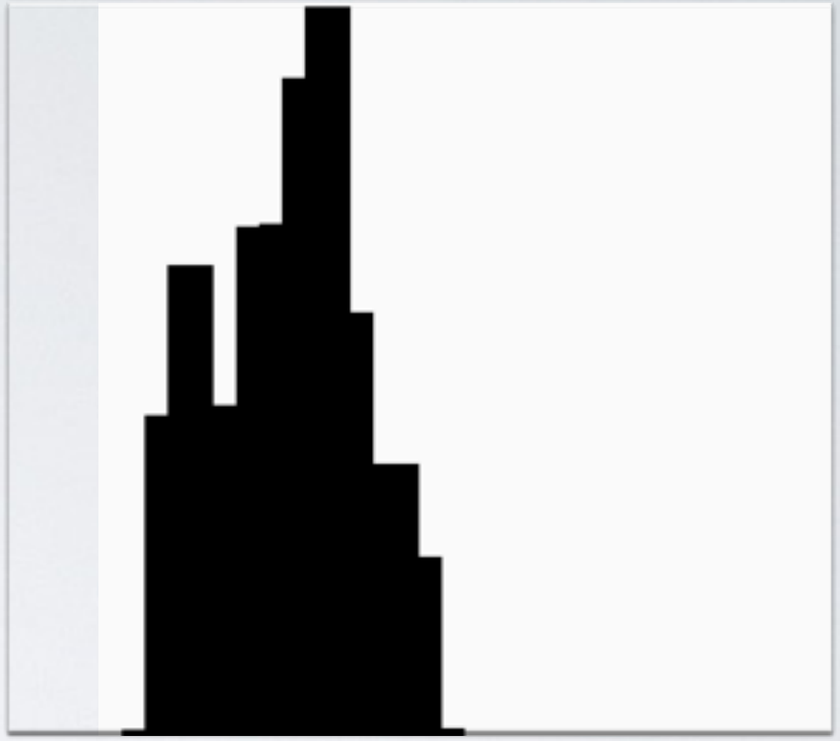
Le temps de calcul variant énormément d'un nombre de clusters à un autre, et selon les centres initiaux, nous avons essayé de créer une base de donnée recensant tous les essais effectués.

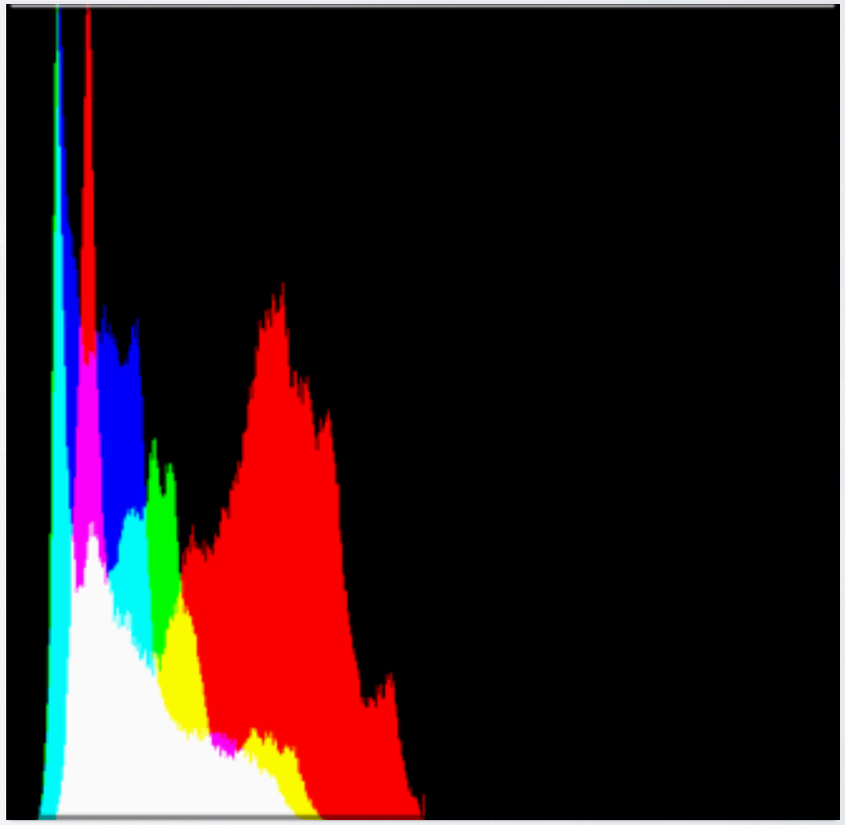
K-MEANS **500** CLUSTERS

OPTIONS

Seuillage damier (x:90 y:90 s :150)

histogramme IMNB





Histogramme IMC