Laura Dietsch
Alexis Bihan

Projet C : Minigimp

IMAC 1
2018 - 2019

Sommaire

Liste du travail effectué 2

Histogramme 3

Elaboration et codage des LUT et transformations : 4

Ouverture, sauvegarde d'un fichier PPM: 6

Conclusions individuelles 6

Liste du travail effectué

LUT demandés			
LUT	Fait	Fait mais ne fonctionne pas	Non Fait
Augmentation de la luminosité	**Oui**		
Diminution de la luminosité	**Oui**		
Augmentation du contraste	**Oui**		
Diminution du contraste	**Oui**		
Inversion de la couleur	**Oui**		
Effet sépia	**Oui**		

Autres LUT et transformations d'images		
Nom	Description	Fait
noir et blanc	Cet effet passe l'image en niveaux de gris	**Oui**
une couleur	Cet effet crée une image en nuances d'une couleur (rouge, vert, bleu, jaune, violet, orange et rose)	**Oui**
rouge et bleu selon intensité	Cet effet rend les nuances sombres bleu et les nuances claires rouge (on pourrait faire avec d'autres couleurs mais cela n'a pas été implémenté)	**Oui**
filtre bicolor	Cet effet applique un dégradé linéaire entre 2 couleurs au choix	**Oui**
nuances de gris sauf une couleur (1)	Cet effet passe l'image en nuances de gris sauf pour une couleur qui garde son intensité originale (rouge, vert ou bleu)	**Oui**
nuances de gris sauf une couleur (2)	Cet effet passe l'image en nuances de gris sauf pour une couleur qui se retrouve boostée à 255 (rouge, vert ou bleu)	**Oui**
symétrie verticale	Miroir de l'image selon la symétrie verticale	**Oui**
symétrie horizontale	Miroir de l'image selon la symétrie horizontale	**Oui**
symétrie complète	Miroir complet de l'image (horizontal puis vertical)	**Oui**
flou	Rend l'image floue	**Non** mais réfléchi (voir commentaires code)
historique des transformations subies	Écrit dans le header de l'image, l'historique des transformations subies par celle-ci si elle en a subi	**Non**

Histogrammes	
Histogramme en nuances de gris	Analyse une image et crée un fichier ppm de l'histogramme de nuances de gris correspondant. L'utilisateur peut choisir la hauteur et la largeur de l'histogramme.
Histogramme RVB	Analyse une image et crée un fichier ppm de l'histogramme des couleurs RVB correspondant. L'utilisateur peut choisir la hauteur et la largeur de l'histogramme.

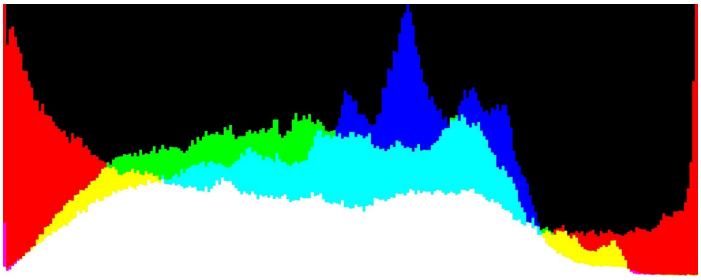
Histogramme

L'analyse de l'image pour créer l'histogramme a été facile à coder. Cependant la création de l'image comportant l'histogramme a été complexe. On s'est proposé de créer une fonction créant l'histogramme qui prend en paramètres un pointeur sur l'image à analyser, la hauteur et la largeur voulues de l'image de l'histogramme.

On a d'abord voulu créer une image avec une seule boucle, pixel par pixel. Mais cela était trop complexe car on ne savait pas facilement où on est placé dans l'image sur les axes x et y. Donc nous avons créé deux boucles imbriquées, la première parcourant l'axe x et la seconde l'axe y. Cela permet de raisonner plus clairement puisque qu'on complète des colonnes pixel par pixel.

Un problème a été de définir la hauteur de l'histogramme. En effet, si on garde les données brutes de l'histogramme sans les mettre à l'échelle, les colonnes seront plus grandes que la hauteur de l'image et on aura une représentation erronée. On a donc créé une variable stockant la valeur maximale de l'histogramme et on la met à jour à chaque itération de l'analyse de l'image. Lors de la création de l'image de l'histogramme on la met à l'échelle en créant un coefficient de proportionnalité : (hauteur de l'image / valeur maximale).
On met aussi à l'échelle la largeur avec un coefficient de 255/(largeur-1) car nos données sont stockées pour les nuances de 0 à 255.

Pour cette image voici les histogrammes en nuances de gris et en RVB obtenus à l'aide de notre programme.



En analysant une image composée essentiellement de rouge on s'est aperçu que toute la couleur verte est presque absente de l'image ce qui crée des pics près de 0. Cela écrase le reste de l'histogramme on a donc décidé de ne pas prendre en compte les 5 premières et 5 dernières colonnes de l'histogramme dans le calcul de la valeur maximale de l'histogramme. La hauteur de l'histogramme est ainsi calculée en fonction d'une valeur plus significative vis à vis du reste de l'image et l'histogramme est devient plus lisible.

Image analysée

Histogramme en mettant à jour la valeur max en analysant toutes les nuances de couleur

Histogramme sans analyser les 5 premières et 5 dernières nuances de couleur

Elaboration et codage des LUT et transformations :

▸ La notion de LUT nous a posé quelques problèmes au cours de ce projet. Nous avons, à tort, privilégié l'action et la réflexion et nous sommes tout de suite lancé dans la création et l'application de LUT à nos images. Cela a eu pour résultat la création de fonctions recopiant entièrement l'image pour changer les valeurs des composantes de couleur rouge, vert et bleu de chaque pixel de celle-ci.

Nous ne nous sommes pas rendu compte de notre erreur avant quelques temps comme les transformations fonctionnaient. C'est en discutant avec d'autres groupes que nous avons remarqué qu'ils avaient créé une structure pour les LUT prenant la forme de 3 tableaux d'entiers de 256 valeurs. Un pour chaque couleur. Après avoir constaté cette différence, nous nous sommes replongés dans le sujet du projet et avons étudié plus en profondeur la notion de LUT. Nous avons alors compris qu'un LUT était une table de correspondance indiquant la nouvelle valeur d'intensité pour une composante de valeur en fonction de sa valeur originale. Cette façon de faire est bien plus économe et efficace que ce que nous avions fait car on peut appliquer un même LUT à plusieurs images sans avoir besoin de les recopier complètement.

Une fois la notion comprise, nous avons créé la structure LUT et adapté nos fonctions pour que celles-ci agissent non plus sur l'image mais sur le LUT. Heureusement pour nous, cela nous a pris peu de temps et nous n'avons pas rencontré de problèmes particuliers pour modifier notre code.

▸ L'effet sépia nous a causé bien du soucis. En effet, selon la définition de l'effet trouvée sur Internet un sépia est défini tel que :

*outputRed = (inputRed * .393) + (inputGreen *.769) + (inputBlue * .189)*
*outputGreen = (inputRed * .349) + (inputGreen *.686) + (inputBlue * .168)*
*outputBlue = (inputRed * .272) + (inputGreen *.534) + (inputBlue * .131)*

Chaque couleur d'arrivée a besoin des valeurs des trois composantes de couleur du pixel avant transformation. Or, les effets que nous avions codé avant de faire le sépia ne tombaient pas dans cette catégorie d'effets car les nouvelles valeurs de rouge, vert et bleu ne dépendaient pas de chacune des anciennes. Nous nous sommes donc heurtés à un problème : "Comment réaliser un beau LUT sépia ?"

Nos premiers essais essayant d'appliquer la définition du sépia à notre image ont donné ce résultat:

Après avoir beaucoup réfléchi et essayé, nous avons conclu qu'il était impossible d'obtenir un effet sépia correct général à appliquer à chaque image. En effet, un beau sépia dépend de chaque pixel de l'image c'est pour ça que nous avons décidé de transformer l'image afin d'obtenir cet effet. Nous passons tout d'abord l'image en noir et blanc et appliquons ensuite la définition plus haut pour créer un LUT sépia spécifique à l'image.

▸ Les effets miroir nous ont aussi donné un peu de fil à retordre. Cette fois, le problème ne relevait pas de la compréhension de la mission mais plutôt de la réalisation. En effet, la structure du tableau de pixels ne rend pas facile l'inversion de ceux-ci. Nous nous sommes rapidement rendu compte qu'en inversant le premier élément du tableau d'une ligne de l'image avec le dernier, nous échangions le rouge avec le bleu. Nous avons donc passé un petit moment à essayer de faire fonctionner ces échanges de pixels correctement pour faire correspondre le rouge au rouge et le bleu au bleu.
L'effet miroir vertical a particulièrement été compliqué car il faut savoir exactement où se trouve le milieu de l'image pour arrêter d'échanger les pixels par rapport à l'axe symétrique. De plus, pour pouvoir échanger les valeurs correctement, il faut créer un second index qui parcourt la ligne dans l'autre sens.

En résumé, les principaux problèmes que nous avons rencontré dans cette partie n'étaient pas d'ordre algorithmique (sauf peut-être pour les effets miroir) mais plutôt de l'ordre de la compréhension du problème à résoudre et de la tâche à effectuer.

Ouverture, sauvegarde d'un fichier PPM:

C'est très certainement l'une des parties qui nous aurait posé le plus de problèmes et pris le plus de temps si on ne nous avait pas donné le code pour le faire. En effet, une fois capables d'ouvrir un fichier PPM et de le sauvegarder il ne restait plus qu'à faire l'histogramme d'une image et à créer des LUT.

Conclusions individuelles

Laura : Ce qui m'a posé le plus de problèmes c'est de conceptualiser sur papier les algorithmes de l'histogramme. La partie analyse de l'image était assez simple mais la partie création des images représentatives des histogrammes était difficile. J'ai fait, pendant plusieurs heures, des schémas sur papier avec des petits tableaux de pixels pour comprendre comment j'allais les parcourir et les remplir de données RGB à partir des données récoltées pendant l'analyse. Une fois que j'ai eu compris comment cela allait fonctionner le code et le débugage ont été plus rapides et plus faciles pour moi. C'est donc sur la partie algorithmique pure que j'ai progressé grâce à ce projet.

Alexis : Je me suis principalement concentré sur l'élaboration des LUT au cours de ce projet. Comme à mon habitude, j'ai passé trop peu de temps à étudier le sujet, réfléchir au concept et à ce qui était demandé pour privilégier le code et l'action. C'est une erreur qui m'a coûté un certain temps plus tard dans le projet, lorsque je me suis aperçu que je n'avais pas implémenté correctement la notion de LUT. J'ai dû tout arrêter pour reprendre depuis la base et repenser tout ce que j'avais déjà codé. Après être revenu sur le concept d'un LUT et l'avoir apprivoisé, j'ai pu repartir sur une base saine et adapter mon code. A partir de ce moment là je pensais que tout serait plus simple mais il n'en a pas été ainsi. J'ai en effet rencontré un problème lors de la création de l'effet sépia. Je ne parvenais pas à comprendre en quoi il était pertinent de créer un LUT sépia au vu de la définition de cet effet et de sa spécificité à chaque image. On peut donc dire que ce qui m'a posé le plus de problèmes dans ce projet c'est l'appropriation du sujet et des différentes notions impliquées. Je n'ai, au contraire, pas rencontré de problèmes algorithmiques particuliers ou très difficile à résoudre donc une fois le problème des LUT surmontés tout s'est bien déroulé.

J'ai beaucoup aimé travailler sur ce projet qui nous a permis de travailler et de manipuler l'image d'une façon nouvelle pour moi et ça a été très instructif. Si je devais retenir une chose de mes erreurs, c'est que je me lance très souvent trop précipitamment dans le code sans prendre le temps de réfléchir auparavant. J'essaierai d'être plus vigilant à l'avenir.