Université Bordeaux 1 Licence Sciences et Technologies

Image et son : TD 1

Traitement d'images

L'objet de ce TP est de manipuler les images avec GIMP et à l'aide de code en langage C. Vous trouverez des images test à l'adresse `http://dept-info.labri.fr/ENSEIGNEMENT/imageson/IMAGE/TD1/`.

GIMP

GIMP (GNU Image Manipulation Program, `http://www.gimp.org/`) est un outil de composition et de traitement d'images.

Exercice 1

1 - Lancez GIMP et créez une nouvelle image. Identifiez/utilisez les outils de dessin accessibles au travers des icônes de la fenêtre principale.

2 - Créez une image `degrade_gimp.pgm` qui représente un dégradé linéaire horizontal du noir vers le blanc.

Format PGM et PPM

Une image PGM (Portable Grey Map) est une image en niveaux de gris : à chaque pixel est affectée une intensité correspondant au niveau de gris du pixel. La valeur nulle correspond au noir, l'intensité maximale correspond au blanc.

Une image PPM est une image couleur : à chaque pixel est affectée une couleur représentée par ses trois composantes rouge, verte et bleue. Par exemple, le pixel de couleur (255, 0, 0) sera rouge vif.

Un fichier représentant une image au format PGM/PPM devra contenir, dans l'ordre :
- $P2$ pour le format PGM, $P3$ pour le format PPM,
- la hauteur et la largeur de l'image,
- La valeur maximale que peut prendre une intensité (255 en général),
- le niveau de gris de chaque pixel si $P2$, la valeur de chacune des composantes de la couleur du pixel si $P3$. Les pixels sont stockés de gauche à droite et de haut en bas.

Exercice 2

1 - Ouvrez avec Emacs le fichier `image1.pgm`. Repérez l'emplacement des sections ci-dessus, puis modifiez la couleur des pixels supérieur gauche et inférieur droit. Visualisez cette modification avec GIMP (faites un zoom si nécessaire).

2 - Faites de même avec `fond.ppm`

3 - Pour charger et enregistrer des images aux formats PGM/PPM, nous utiliserons le fichier `images.c`. Analysez le code source et plus particulièrement les fonctions `lireNdgImage` et `ecrireNdgImage`. Compilez et exécutez le programme. Que fait-il ?

4 - Ajoutez une instruction affectant au pixel en haut à gauche de l'image la couleur noire. Vérifiez sous GIMP que votre modification est prise en compte et que l'image est bien enregistrée.

Exercice 3 *Création d'images PGM*

1 - Ecrivez et testez une fonction `void imageUnie(char* nom, int ng)` qui crée et enregistre dans le fichier `nom` une image unie de niveau de gris ng. Créez ainsi une image unie (`gris.ppm`) de niveau de gris 50 en rajoutant dans le `main` un appel à cette fonction.

2 - Ecrivez et testez une fonction `void degradeHorizontal(char* nom, int ng1, int ng2)` qui crée et enregistre dans le fichier `nom` une image comportant un dégradé linéaire horizontal du niveau de gris $ng1$ au niveau de gris $ng2$. Créez ainsi une image unie (`degrade.pgm`) avec $ng1 = 100$ et $ng2 = 200$.

Images PPM

Exercice 4

1 - Ouvrez avec GIMP `image4.ppm` et sauvegardez le fichier en tant que `image5.pgm`. Ouvrez avec Emacs ces deux fichiers et observez les valeurs obtenues.

2 - En vous inspirant du code contenu dans le fichier `images.c` et de la définition du type, écrivez une fonction `lireCoulImage (char *nom, coulIm im)` et une fonction `ecrireCoulImage (char *nom, coulIm im)` ouvrant et sauvegardant des images couleurs.

3 - Chargez l'image couleur `fleur.ppm` puis modifiez la couleur du pixel se trouvant en son milieu (position (LARGEUR/2, HAUTEUR/2)), il doit apparaitre en vert. Sauvegardez l'image sous un nouveau nom (par exemple `test.ppm`) et vérifiez votre résultat.

Espaces colorimétriques

Exercice 5 Conversion couleur/niveau de gris

1 - Ecrivez une fonction `RGBToGray (coulIm src, ndgIm dest)` transformant une image couleur en niveaux de gris. Vous utiliserez la transformation suivante :

$$Y = 0.3R + 0.59G + 0.11B$$

Testez avec l'image `fleur.ppm`. Comparez votre résultat avec celui de GIMP (`Image/Mode/Niveaux de gris`).

Exercice 6 Exercice complémentaire : Conversion RGB/YCbCr

Cet exercice est à traiter après tous les autres.

1 - Ouvrir l'image `fleur.ppm` dans GIMP. Observez les trois canaux de couleur Rouge Vert Bleu à l'aide de l'outil `Couleurs/Composants/Décomposer` et en affichant les calques (Ctrl+l).

2 - L'espace de couleur YCbCr est un espace luminance/chrominance utilisé pour le codage des couleurs dans les systèmes de télévision notamment. Il existe différentes matrice de transformation de RGB vers YCbCR. Vous utiliserez la suivante :

$$Y = 0.299R + 0.587G + 0.114B$$

$$C_b = -0.1687R - 0.3313G + 0.5B + 128$$

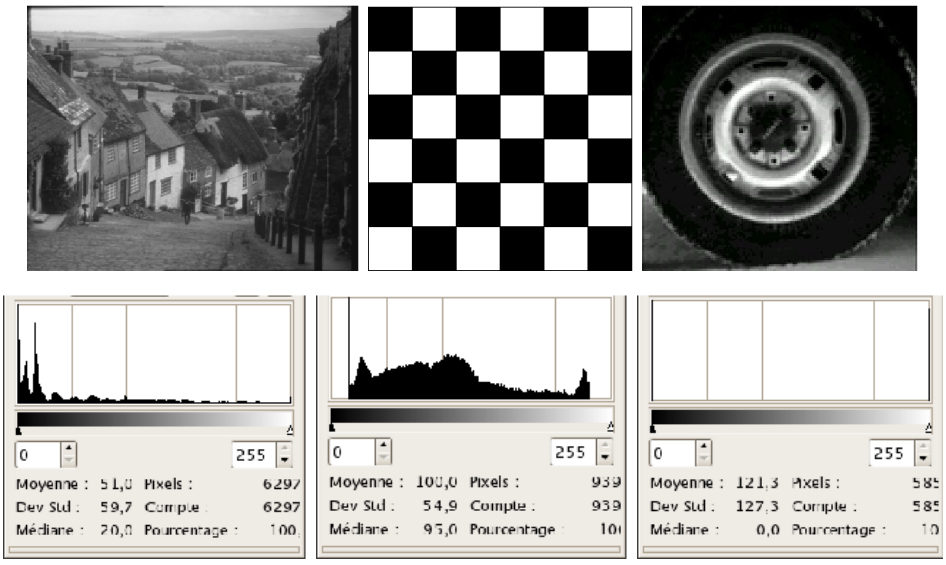
$$C_r = 0.5R - 0.4187G - 0.0813B + 128$$

Ecrivez une fonction `RGBToYCbCr (coulIm src, ndgIm Y, ndgIm Cb, ndgIm Cr)` transformant une image RGB en trois images en niveaux de gris, représentant les trois canaux Y, Cb et Cr. Testez avec les images `fleur.ppm` et `lena.ppm`. Comparez votre résultat avec celui de GIMP.

Histogramme

Exercice 7

1 - Dans la figure ci-dessous, associer à chaque image son histogramme. Expliquez.



2 - Ouvrez les images `degrade.pgm`, `image1.pgm` et `image2.pgm`. Sur chacune de ces trois images, observez et justifiez l'histogramme (menu `Couleurs/Informations/Histogramme`).

Exercice 8

1 - Ajoutez au début de votre code la définition du type Histogramme : `typedef float Histo[256]`.

2 - Implémentez la fonction `void remplir_histogramme (ndgIm im, Histo h)` remplissant un histogramme à partir d'une image PGM.

3 - Utilisez la fonction `void histogramme_dat(char* hnom, histo h)` pour écrire les valeurs de l'histogramme dans le fichier `nom` (d'extension `.dat`). Vous tracerez ensuite l'histogramme en utilisant gnuplot (`http://www.gnuplot.info/`). Vous testerez sur les images `degrade.pgm`, `image1.pgm` et `image2.pgm`.

4 - Comparez vos résultats avec l'histogramme proposé par GIMP.

5 - Ecrivez ensuite une fonction permettant de normaliser un histogramme afin que la somme de toutes les valeurs soient égales à 1.