Omnes
Nathanaël

Compte-rendu
TP2

2 - Définition d'une classe pour représenter des images arbitraires

J'ai bien créé une classe *Image2D* basée sur la classe *GrayLevelImage2D* du TP précédent. La question a été traitée complètement : toutes les méthodes nécessaires ont été implémentées et le code de test est fonctionnel. Le programme de test s'appelle *testGrayLevelImage2DPartie2*.

3 - Introduction des images couleurs

La classe Color est fonctionnelle, le programme de test est appelé *testColorImage2DPartie3*, il affiche une matrice 3x3 composée de couleurs définies sur 3 canaux (rouge=125, vert=255, bleu=0).
La matrice attendue dans le terminal est donc :

```
125 255 0 125 255 0 125 255 0
125 255 0 125 255 0 125 255 0
125 255 0 125 255 0 125 255 0
```

4 - Premier itérateur sur images quelconques

Les itérateurs (Iterator et ConstIterator) fonctionnent correctement, une fois exécuté, le programme nommé *testColorImage2DPartie4* génère deux fichiers de sortie :

- OutputColorsIter.ppm généré par l'itérateur non constant
- OutputColorsConstIter.ppm généré par l'itérateur constant

5 - Un importeur / exporteur PBM générique

Les importeur ont été implémentés dans le fichier *Image2DReader.hpp* et les exporteurs sont dans *Image2DWriter.hpp*. Le programme de test pour importer et exporter une image en niveaux de gris s'appelle *testGrayLevelImage2DPartie5*, comme dans le premier TP, l'image est éclaircie.

Le programme pour l'import/export de fichiers *.ppm* s'appelle *testColorImage2DPartie5*, elle fonctionnait lors de sa première implémentation mais pour une raison inconnue, sur certains fichiers (comme *lena-color.ppm*) les couleurs sont inversées. De plus, certaines images sont

bien cadrées et d'autres on leur cadre déplacé. J'ai essayé avec les fichier *lena-color.ppm, kowloon.ppm* et *papillon.ppm*.

6 Premier test: on inverse les canaux rouge et bleu

Le programme *invert_red_blue* sert à inverser le canal rouge et le canal bleu de tous les pixels d'une image. Il fonctionne correctement avec toutes les images .ppm.

7 On rajoute les accesseurs et un itérateur générique

Le programme s'appelle *save-green-channel.* A la ligne 42, j'ai dû modifier *GenericConstIterator* en *GenericIterator* suite à une erreur quand je lance le programme. Je n'ai donc pas complètement réussi à le faire fonctionner. Cependant, il marchait correctement avant que j'implémente le *GenericIterator*.

8 Créer les nouveaux accesseurs à la composante rouge et bleue

Le programme s'appelle *save-channels*. Comme le programme de la partie 7, celui-ci fonctionnait correctement avec *GenericConstIterator* jusqu'à l'implémentation d'un itérateur non constant. Mais après la modification de *GenericConstIterator* en *GenericIterator* alors le programme fonctionne.

9 Itérateurs génériques non constants

Le programme s'appelle *cathode-ray-effect*, il fonctionne correctement avec les images *.ppm* de test (*lena-color.ppm* et *papillon.ppm*).

10 Espace TSV (HSV) et histogramme d'une image couleur

Le programme s'appelant *histogramme* fonctionne correctement. Cependant, je ne suis pas certain d'avoir utilisé la classe *Histogramme* comme vous l'attendiez même si les résultats sont similaires.

Vous pouvez décommenter les lignes 16-17 d'*Histogramme* pour afficher les valeurs saisies dans le terminal.

11 Égalisation d'image couleur

Le programme d'égalisation s'appelle *egaliseur-couleur*, il ne fonctionne pas tout à fait, les couleurs de l'image de sortie ne sont pas du tout celles attendues. La méthode setHSV doit avoir une partie manquant mais je n'ai pas réussi à trouver laquelle.
L'histogramme cumulé est bien rempli et la méthode *Egalisation* doit être la même que celle du TP précédent. Peut-être que mon erreur vient d'une approximation de la variable V à un moment.

12 Un peu d'imagination

Deux programmes sont présents :

- *filtre-sepia-graylevel*, il prend en premier argument une image *.pgm* et sort une image *.ppm*, le deuxième argument doit être du type *<nomFichier.ppm>*
- *filtre-sepia-color*, prend en premier argument une image *.ppm* et sort une image du même type, le second argument doit aussi être un nom avec l'extension *.ppm*.

Ces programmes transforment soit une image couleur, soit une image en niveaux de gris en la même image avec un filtre sépia appliqué.

Les algorithmes appliqués sont disponibles dans la classe *Image2D.hpp*, lignes 176, 195 et 212.