TP Interopérabilité - Manipulation d'une image au travers de SQL

Dans ce TP, nous allons appliquer le filtre de Marr et Hildreth via des requêtes SQL H2 et en utilisant des fonctions C/C++ et Python. Votre programme principal sera écrit en Java, dans un seul fichier, avec au moins une méthode statique par question (`exercice1`, `exercice2`...).

Vous pouvez vous baser

- sur les cours et sur les TP effectués précédemment
- sur les TP effectués dans le cours Introduction à l'Informatique Graphique (https://community-sciences.unilim.fr/course/view.php?id=1901&section=1#tabs-tree-start) pour les opérations sur les images, et sur les images du projet de démarrage du premier semestre (`hand.jpg`, `building.png`)

Partie 1: génération de données pour SQL

Exercice 1: table virtuelle GAUSSIENNE

Programmez une table virtuelle SQL `GAUSSIENNE(k)` renvoyant une table ayant 3 colonnes, `x`, `y` (entiers), et `valeur` (flottant ou double). `k` est la taille de la gaussienne et est toujours impaire. Si `k` (entier) vaut 3, `x` et `y` ont des valeurs entières allant de -1 à 1 (-1, 0, 1 pour x et y) ; si `k` vaut 5, les valeurs vont de -2 à 2.

Affichez les valeurs de cette table dans la console pour des valeurs de k de 5 et 7.

Exercice 2: table virtuelle RGBIMAGE

Vous allez avoir besoin de lire des images depuis des fichiers disque. Programmez une table virtuelle SQL `RGBIMAGE(path)` renvoyant une table ayant 5 colonnes, `x`, `y`, `r`, `g`, `b` (tous entiers). `r`, `g`, `b` ont des valeurs entre 0 et 255 et représentent les intensités des pixels de coordonnées `x`, `y` des pixels de l'image `path` sur disque. `path` est résolu comme étant un chemin relatif à votre workspace, avec les fichiers images résidant dans votre workspace.

Notez que vous pourrez créer des images noires dans le code du TP en vous appuyant sur `SYSTEM_RANGE`:

```
CREATE TABLE UNERGBIMAGENOIRE(LARGEUR INT, HAUTEUR INT) AS SELECT ... FROM
SYSTEM_RANGE(1, ?)
```

`SYSTEM_RANGE` est une table virtuelle intégrée à H2 disposant d'une colonne `X`, et renvoyant des valeurs entre `1` et `?`, où `?` est un paramètre.

Partie 2: sauvegarde sous forme de fichier image du contenu d'une table nommée depuis C/C++

Avec JNI, liez votre application Java à une bibliothèque C/C++. Basez-vous sur le projet de démarrage C/C++ pour les bibliothèques utilisant CMake.

Côté Java, déclarez une méthode statique `sauveImage` prenant en argument le nom d'une table et un chemin de fichier. Depuis l'implémentation JNI de cette méthode statique, lisez le contenu de cette table en partant du principe que son schéma sera toujours sur 5 colonnes, dans cet ordre: `x`, `y`, `r`, `g`, `b`, tous sous forme d'entiers. Sauvegardez une image PNG correspondant au nom donné. Vous pouvez inférer la taille de l'image en examinant les plages de valeurs de `x` et de `y`.

Soignez vos déclarations de sorte que votre code marche indifféremment sous Linux, Windows ou Mac.

Vérifiez que votre implémentation fonctionne en lisant puis en écrivant la même image, et en vérifiant sa validité en la visualisant.

Pour réaliser cet exercice, utilisez `CImg.h` (voir https://cimg.eu/). Ce fichier header permet de manipuler des images (chargement, création, sauvegarde...) et est portable sur tous les systèmes courant.

```
#include "CImg.h"
using namespace cimg_library;
// Crée une image R,G,B de dimension largeur x hauteur
CImg<unsigned char> img(largeur,hauteur,1,3,0); // 1,3,0: RGB 3 octets
img.save(...);
```

Partie 3: opérations sur les images

Vous pouvez faire des requêtes directes sur des tables dynamiques paramétrées. En revanche, pour pouvoir faire des requêtes `UPDATE`, il vous faut pouvoir déverser le contenu de ces tables dans des tables nommées, comme "instances".

Ainsi, vous pouvez par exemple créer une table `UNEIMAGE` qui est le résultat de la lecture de toute la table `RGBIMAGE`:

```
Statement s = conn.createStatement();
s.execute("CREATE TABLE UNEIMAGE AS SELECT * FROM RGBIMAGE(?)");
```

Exercice 3: conversion en niveaux de gris

Effectuez une requête `UPDATE` permettant de convertir l'image de la main (fournie, hand.jpg) en niveaux de gris en utilisant la formule vue au premier semestre (pondérations RGB de valeurs 0.3, 0.59 et 0.11).

Sauvez l'image résultat sur disque `niveauxdegris.png`.

Exercice 4: appliquer un filtre gaussien sur l'image

Appliquez maintenant un filtre gaussien de taille `k` à l'image du batiment (building.png) en utilisant SQL et la table `GAUSSIENNE`. Réutilisez la formule de la gaussienne vue au premier semestre (TP1, Introduction à

l'Informatique Graphique).

Considérez les possibilités de jointure offertes par SQL pour réaliser cette opération.

Isoler le code appliquant la gaussienne dans une méthode statique `gaussienne`, paramétrée par un nom de table (sur laquelle appliquer la gaussienne) et une taille `k`.

Sauvez l'image résultat sur disque pour k=11, `gaussienne11.png`.

Exercice 5: filtre de Marr et Hildreth, soustraction d'images

A présent, fabriquez deux images filtrées avec deux gaussiennes de taille différentes (7 et 5), avec deux requêtes `UPDATE`, représentant des images abritées dans deux fichiers, gaussienne11.png et gaussienne9.png, la première gaussienne étant plus forte que la seconde.

Fabriquez une troisième image marrhildreth.png, étant le résultat de `gaussienne7 - gaussienne5`.

Vous pouvez changer les tailles des gaussiennes, 7 et 5, pour produire de meilleurs résultats si vous le souhaitez. Dans ce cas, adaptez le nom des fichiers en conséquence.

Exercice 6: Seuillage

Transformez l'image en niveau de gris obtenue précédemment en image en noir et blanc en effectuant un seuillage, toujours avec SQL. Isolez votre code dans une fonction `seuillage`.

Soignez bien la valeur de votre seuillage pour que les contours ressortent à la fois le mieux possible, et sans produire trop de pixels parasites (c'est un compromis que vous avez la responsabilité de trouver).

Sauvez l'image résultat et nommez-là `seuillage.png`.

Effectuez maintenant un seuillage par hystérésis. Un seuillage par hystérésis est un seuillage qui sélectionne les pixels si leur intensité

- est au-dessus d'un seuil haut (s_h)
- est comprise entre un seuil bas (s_b) et un seuil haut et qu'un pixel immédiatement voisin a une intensité au dessus d'un seuil haut

Sauvez l'image résultat et nommez-là `seuillagehysterisis.png`. Là encore, soignez les valeurs de s_h et s_b.

Exercice 6: implementation de la gaussienne en Python

Proposez une autre implémentation, `GAUSSIENNEPYTHON`, de sorte que, depuis votre code Java, du code Python soit appelé, et en déléguant les valeurs du calcul de la gaussienne à ce code.

Sauvez les résultats produits, `gaussiennepython7.png`, `gaussiennepython5.png` et `marrhildrethpython.png`.

Exercice 7: downscaling avec antialiassage

Nous allons réaliser une mise à l'échelle vers le bas de l'image seuillée par hystérésis produite dans l'exercice 5.

Le downscaling est le redimensionnement dans une résolution inférieure. Si une image est de taille 32x32, un downscaling de 2 génèrera une image 16x16. Un downscaling de 3 produit au choix une image 10x10 ou 11x11.

Implémenter une fonction `downscale` en Java, prenant en paramètre un chemin d'entrée, de sortie et de taille du downscaling (2, 3, 4...). Effectuez le downscaling avec SQL.

Prenons un exemple avec un downscale de 2. 4 pixels noirs produisent du noir, 4 blancs produisent du blanc, et les autres situations produisent des niveaux de gris selon la quantité plus ou moins grande de noir ou de blanc.

Sauvez l'image résultat avec k=4 (antialiassage.png).