Atelier Indexation d'images

M2 SIC Pro
Ghiles Mostafaoui, Dan Vodislav

1 Introduction

Nous souhaitons réaliser un système capable de répondre des requêtes de recherche dans une bases d'images indexées par le contenu. A partir de bases d'images fournies, il s'agira dans un premier temps d'extraire des caractéristiques spécifiques au contenu de chaque image. Ces caractéristiques seront ensuite stockées dans la base de données afin d'indexer cette base d'images et de développer des requêtes.

1.1 Vue d'ensemble

Le but de l'atelier consiste à produire :

- un outil d'extraction de différentes caractéristiques images (gradients, couleur, histogrammes etc.)
- une base de données ORACLE dans laquelle les images et leurs caractéristiques telles que décrites ci-dessus, ainsi que la signature telle qu'elle est calculée par ORACLE seront stockées.
- des réponses à des requêtes sur les images stockées dans la base.

1.2 Constitution des équipes

Vous devrez vous regrouper par binômes. Chacun pourra ainsi travailler et avancer sur chacune des parties principales de cet atelier à savoir : le traitement d'image et les bases de données.

1.3 Déroulement et notation

L'atelier se déroule sur quatre journées pleines pendant lesquelles les binômes développent à temps plein leurs outils. Bien que le sujet soit présenté sous la forme séquentielle de deux parties distinctes (Partie I et Partie II) il est recommandé aux binômes de veiller à un bon découpage des lots de travail de manière à atteindre leurs objectifs. Un encadrant de la partie I sera présent le premier jour, et un encadrant de la partie II le deuxième jour, les deux derniers jours étant dédiés au travail en autonomie et à la soutenance.

La notation se fera sur la base d'une démonstration par binôme, ainsi que d'un rapport de 5 pages maximum décrivant le dispositif, ses fonctionnalités et l'organisation de l'équipe.

2 Partie I - Ghilès Mostafaoui

L'objectif est ici de vous initier à l'extraction de caractéristiques images simples en langage C. Cet atelier vous permettra, en premier lieu, de mettre en pratique les algorithmes "classiques" vus en cours de remise à niveau (filtrage, calcul du gradient etc.). L'extraction des caractéristiques étant "off line", on ne sera pas ici limités par des contraintes temps-réel (pour cette partie !!) vous permettant ainsi de pouvoir utiliser des algorithmes différents indépendamment de la complexité calculatoire (toute mesure gardée bien entendu...).

L'atelier Traitement d'Images qui suivra celui-ci abordera en revanche les aspects temps-réel en partant de l'acquisition du flux image au traitements temps réel.

2.1 La bibliothèque NRC(Numerical Recipes in C)

Nous utiliserons dans le cadre des travaux pratiques de ce module la bibliothèque Numerical Recipes (NRC) qui contient un ensemble de fonctions qui permettent :

- de gérer les entrées/sorties : lire et écrire des images de type PGM ou PPM (voir le fichier nrio.c)
- de gérer les allocations mémoire : allouer en mémoire des matrices de différents types et de différentes taille (voir le fichier nralloc.c)
- de réaliser des calculs vectoriels et matriciels (voir le fichier nrarith.c)

Pour utiliser cette bibliothèque il suffit d'inclure dans votre fichier ".c" les différents Headers : "def.h", "nrio.h", "nralloc.h" et "nrarith.h".
Nous utiliseront principalement les fonctions suivantes :

- **byte ** LoadPGM_bmatrix(char *filename, long *nrl, long *nrh, long *ncl, long *nch)** : Lecture d'une image de type PGM
- **rgb8 ** LoadPPM_rgb8matrix(char *filename, long *nrl, long *nrh, long *ncl, long *nch)** : Lecture d'une image de type PPM
- **void SavePGM_bmatrix(byte **m, long nrl, long nrh, long ncl, long nch, char *filename)** : Ecriture d'une image de type PGM
- **void SavePPM_rgb8matrix(rgb8 **m, long nrl, long nrh, long ncl, long nch, char *filename)** : Ecriture d'une image de type PPM
- **byte** bmatrix(long nrl, long nrh, long ncl, long nch)** : Allocation d'une matrice de "byte"
- **rbg8** rgb8matrix(long nrl, long nrh, long ncl, long nch)** : Allocation d'une matrice de "rgb8"
- **int** imatrix(long nrl, long nrh, long ncl, long nch)** : Allocation d'une matrice de "int"
- **void free_bmatrix(byte **m, long nrl, long nrh, long ncl, long nch)** : libération de la mémoire pour une matrice de type "byte"
- **vois free_rgb8matrix(rgb8 **m, long nrl, long nrh, long ncl, long nch)** : libération de la mémoire pour une matrice de type "rgb8"
- **void free_imatrix(int **m, long nrl, long nrh, long ncl, long nch)** : libération de la mémoire pour une matrice de type "int"

2.2 Prise en main : Lecture et écriture d'une image de type PGM

Ecrivez votre premier programme que vous nommerez tp1.c qui lit une image pgm qui vous sera fournie et qui enregistre la même image sous le nom "imageTest.pgm".
Afin de compiler votre programme il suffit d'inclure tous les .c comme suit :

gcc -o tp1 tp1.c nrio.c nralloc.c nrarith.c

Note : Pensez à bien libérer vos mémoires !

2.3 Produit de convolution et Filtrage passe bas

Nous voulons ic réaliser un filtrage passe bas (filtre moyenneur) d'une image en niveau de gris. Vous testerez en premier lieu sur l'image "Cubesx3.pgm" avant d'essayer sur d'autres images à choisir parmi les images fournies ou à télécharger sur le net.

Le résultat de notre traitement (filtrage) sera une nouvelle image $g(x, y)$ égale au produit de convolution entre l'image d'origine $f(x, y)$ et la réponse impultionnelle du filtre $h(x, y)$:

$$g(x, y) = f(x, y) * h(x, y) \ .$$

Pour rappel :

$$g(x, y) = \int_{-\infty}^{\infty} \int_{\infty}^{\infty} f(x - u, y - v) \cdot h(u, v) du dv$$

Vous testerez le produit de convolutions en filtrant l'image à l'aide du masque (représentant la réponse impultionnelle) $3x3$:

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

. Rappels :

- Attention, dans notre cas, notre signal 2D (image) n'est pas infini, il y aura donc un effet de bord à gérer, le produit de convolution ne pourra être calculé en dehors des bornes de notre signal (image)
- Ne pas oublier que vous travaillez sur des matrices de type Byte (valeurs comprises entre 0 et 255), il faudra faire attention aux overflows et underflows.

2.4 Calcul du gradient

2.4.1 Gradient horizontal

Convoluez l'image à l'aide du masque de Sobel correspondant au gradient horizontal:

$$\begin{matrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{matrix}$$

Vérifiez le résultat (qu'on notera I_x) en le sauvegardant dans un image de type pgm.

2.4.2 Gradient vertical

De la même faon, convoluez l'image à l'aide du masque de Sobel correspondant au gradient vertical :

$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{matrix}$$

Vérifiez le résultat (qu'on notera I_y) en le sauvegardant dans un image de type pgm.

2.4.3 Norme du gradient

Calculez la norme du gradient $\sqrt{I_x^2 + I_y^2}$.

2.5 Détection de contours

Les valeurs hautes de la norme du gradient indiquent le présence de zones de contours. Afin de détecter les contours de l'image, il suffit alors de seuiller la norme calculée précédemment; il en résultera une image binaire (valeurs des niveaux de gris à 0 ou 1) représentant les contours de l'image. Faites plusieurs essais avec différents seuils (sur des images réelles) pour apprécier la difficulté de trouver un bon rapport signal sur bruit.

2.6 Histogrammes

Faire un programme qui calcule l'histogramme en niveaux de gris d'une image et qui sauvegarde ce dernier dans un fichier texte.

2.7 Lecture d'une image couleur

Utiliser la bibliothèque NRC pour lire une image couleur, la transformer en image en niveaux de gris et la sauvegarder en format pgm.

2.8 Utilisation des caractéristiques images

Voici quelques pistes pour utiliser ces différents traitements pour caractériser les images :

- la moyenne de la norme du gradient pour rendre compte de la présence ou pas de zones texturées. Est ce suffisant pour caractériser la texture ?
- le nombre de pixels de contour dans l'image
- le taux de rouge, de vert et bleu dans chaque image. La aussi attention, est ce représentatif ? une image entièrement "blanche" aura un taux max aussi bien en rouge, vert et bleu ....
- la distance entre histogrammes. Vous pouvez essayer une simple différence entre les vecteurs histogrammes puis tester des distances plus élaborées (distance de bhattacharyya par exemple)
- la différenciation images couleur et images en niveaux de gris

IMPORTANT : Comme le traitement se fait sans "grosse" contrainte temps-réel, vous êtes libres d'utiliser des caractéristiques, algorithmes, distances... plus complexes.

3 Partie II - Dan Vodislav

3.1 Rappels

Nous rappelons ci-dessous les éléments vus en M1 à propos de la gestion d'images sous ORACLE.

```
set serveroutput on;
-- declaration de repertoire contenant les images (à faire par le DBA)
create directory IMG as '/home/user/IMG'
grant read on directory images to public

-- creation de la table d'images
create table multimedia(
    nom varchar2(50),
    image ordsys.ordimage,
    signature ordsys.ordimageSignature
);

declare
    i ordsys.ordimage;
    ctx RAW(400) := NULL;
    ligne multimedia%ROWTYPE;
    cursor mm is
       select * from multimedia
       for update;
    sig1 ordsys.ordimageSignature;
    sig2 ordsys.ordimageSignature;
    sim integer;

begin
-- insertion de la premiere image, de contenu vide
insert into multimedia(nom, image, signature)
values ('image1.jpg', ordsys.ordimage.init(), ordsys.ordimageSignature.init());
commit;

-- chargement du contenu de l'image a partir du fichier
select image into i
from multimedia
where nom = 'image1.jpg'
for update;

i.importFrom(ctx, 'file', 'IMG', 'image1.jpg');

update multimedia
set image = i
where nom = 'image1.jpg';
commit;
-- proceder de meme pour les autres images

-- generation des signatures
for ligne in mm loop
    ligne.signature.generateSignature(ligne.image);
    update multimedia
    set signature = ligne.signature
    where current of mm;
endloop;
```

```
commit;

-- on peut generer un index sur les signatures d'images
create index imgindex on multimedia(signature) indextype is ordsys.ordimageindex;
-- test de similarite entre image1.jpg et image2.jpg
select signature into sig1
from multimedia
where nom = 'image1.jpg';

select signature into sig2
from multimedia
where nom = 'image2.jpg';

sim := ordsys.ordimageSignature.isSimilar(sig1, sig2,
    'color = 0.5, texture = 0, shape = 0, location = 0', 10);

if sim = 1 then dbms_output_line('Images similaires')
else dbms_output_line('Images non similaires')
end;
```

3.2 Travail demandé

À partir des résultats obtenus dans la Partie I stockés dans un fichier, on demande d'écrire le code SQL et PL/SQL de façon à répondre aux questions ci-dessous.

1. Créer une table SQL qui stocke les images à analyser, ainsi que leur signature ORACLE et leurs différentes caractéristiques évoquées dans la Partie I.
2. Charger les images, leur signature et leurs caractéristiques dans cette table (ces caractéristiques ont été calculées et sauvegardées dans la partie I).
3. Calculer les identifiants des images similaires à une image de la base spécifiée par son nom.
   (a) En utilisant les descripteurs ORACLE.
   (b) En utilisant les caractéristiques calculées.

   Les différents paramètres nécessaires à la définition de similitude seront explicités et commentés dans le rapport. Commenter les résultats obtenus.
4. Mesurer les temps d'exécution des requêtes et comparer les deux approches. Considérez le cas où vous travaillez seulement sur le jeu initial de 10 images et celui où vous utilisez les deux jeux d'images ensemble. Pour le calcul avec les signatures Oracle, comparez les temps sans ou avec index.
5. En utilisant uniquement les caractéristiques calculées dans la Partie I :
   (a) Afficher les identifiants et noms des images contenant peu ou pas de vert et pour lesquelles le niveau de rouge est important.
   (b) Afficher les identifiants et noms des images noir et blanc.
   (c) Afficher les identifiants et noms des images texturées.
6. Proposer au moins une requête différente de celles demandées ci-dessus et qui vous paraisse pertinente dans le cadre de cet atelier. On fournira bien sûr la (ou les) réponse(s) à cette (ou ces) requête(s) !