Projet Fondements de l'informatique

Luc Brun & Eric Ziad-Forest

1 Informations générales

Début du projet : Semaine du 12 décembre

Remise du projet : Semaine du 13 février

Nombre de personnes : le projet est effectué en binôme.

Documents à remettre : Un rapport écrit de 10 pages (max) avec une description des méthodes algorithmiques et des structures de données utilisées ainsi qu'un jeu d'essais. Ce document doit être remis à votre encadrant de TP de C.

Les fonctions de base permettant de charger/sauver une image et de lire ses valeurs sont disponibles dans le répertoire :

`/home/public/PROJET_FONDEMENTS/MODULE_IMAGE`

Des exemples d'images sont également disponibles sous :

`/home/public/PROJET_FONDEMENTS/IMAGES`

2 Utilisation du module image

Définissez les fonctions suivantes dans un fichier image_util.{c,h} :

1. `extern void draw_square(image,int,int,int,int,unsigned char*)`
   Cette fonction dessine un rectangle dans une image. Les quatre entiers (arguments 2 à 5) correspondent dans l'ordre à $(x_{min}, y_{min}, x_{max}, y_{max})$. Le point (x_{min}, y_{min}) correspond au point en haut à gauche du rectangle et le point (x_{max}, y_{max}) au point en bas à droite. La couleur (ou le niveau de gris) des bords est déterminée par l'argument six.
2. `extern void give_moments(image,int,int,int,int,int*,double*,double*);`
   Cette fonction calcule les moments d'ordre 0,1 et 2 des pixels situées dans un rectangle (cf question précédente). Le moment d'ordre 0 (sixième paramètre) correspond au nombre de pixels dans le rectangle. Le moment d'ordre 1 est la somme des intensités (ou couleurs) des pixels dans le

rectangle. Le moment d'ordre 2 est la somme des carrés des intensités ou couleurs dans le rectangle. On a donc en couleur et dans l'espace RGB :

$$\begin{array}{lcl} M0 & = & \sum_{P\in B} 1 \\ M1 & = & \sum_{P\in B}(R(P),G(P),B(P)) \\ M2 & = & \sum_{P\in B}(R^2(P),G^2(P),B^2(P)) \end{array}$$

où B correspond au rectangle spécifié par les paramètres deux à cinq, $R(P), G(P), B(P)$ correspond aux composantes rouge, vertes et bleu du pixel P. Les équations sont identiques en niveaux de gris. On a simplement dans ce cas une seul composante par pixel. Notez que $M0$ est toujours un scalaire alors que $M1$ et $M2$ sont des vecteurs en couleur et des scalaires en niveau de gris.

3 Le quad tree

Un quad tree ou arbre quaternaire est un arbre dont chaque noeud a quatre fils. On utilise un tel arbre pour représenter une division récursive de l'image. Définissez le type quadtree comme un pointeur sur un struct quadtree. La structure quadtree est définie dans un fichier quadtree.c de la façon suivante :

```
struct quadtree
{
quadtree sons[4];
double M0,M1[3],M2[3];
}
```

Dans le cadre d'une subdivision d'une image chaque noeud de l'arbre est associé à une zone rectangulaire de l'image. Si un noeud admet une subdivision, sa zone est divisée en 4 sous partie en divisant la largeur et la hauteur du rectangle par 2. Chaque sous rectangle(appelé un quadrant) est codé par un fils du noeud. Par convention :

- sons[0] représente le quadrant en haut à gauche,
- sons[1], le quadrant en haut à droite,
- sons[2], le quadrant en bas à droite,
- sons[3], le quadrant en bas à gauche.

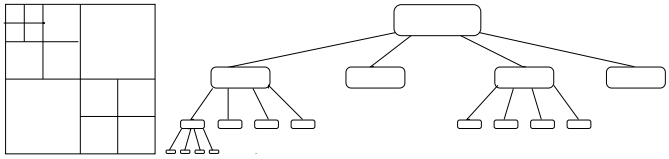
Les moments M0,M1,M2 ne prennent de sens que lorsque le quadtree est associé à une image.

Un exemple de quadtree est disponible sur la figure 1.

3.1 Création, suppression, subdivision

Définissez les fonctions :

```
extern quadtree create_quadtree();
extern void quadtree_subdivide(quadtree);
extern void delete_quadtree(quadtree);
```



(a) Image (b) Arbre

FIGURE 1 – Un exemple de codage d'une subdivision récursive par un quadtree.

La première fonction renvoie simplement un quadtree réduit à un seul noeud (ses fils sont nulls). La seconde fonction, initialise les 4 fils d'un quadtree.

La troisième fonction supprime tous les noeuds d'un quadtree (le noeud lui même et ses fils). On pourra utiliser une fonction récursive remontante.

3.2 Approche descendante

Définissez les fonctions suivantes :

```
extern quadtree split_image(image,double);
extern void draw_quadtree(image,quadtree,unsigned char*);
```

La première fonction crée un quadtree en découpant récursivement les noeuds jusqu'à ce que les zones associées aient une variance inférieure à un seuil (second argument). La variance pour les images en niveaux de gris est donnée par :

$$var = \frac{1}{M0}\left(M2 - \frac{M1 * M1}{M0}\right)$$

Cette fonction initialise les moments de chaque noeud du quadtree à partir du carré correspondant dans l'image. Doit on calculer les moments de tous les fils d'un noeud ?

La seconde fonction dessine dans l'image les bords des noeuds du quadtree qui correspondent à des feuilles. La couleur des bords est passée en troisième argument.

3.3 Approche mixte

Définissez les fonctions suivante :

```
extern quadtree create_default_quadtree(int);
extern void init_quadtree(quadtree,image)
extern void update_quadtree(quadtree,image,double)
```

— La première fonction crée un quadtree dont toutes les feuilles ont une même hauteur h.
— La seconde fonction initialise les moments des feuilles d'un quadtree à partir de l'image puis en déduit les moments de tous les noeuds.

— La troisième fonction supprime l'ensemble des fils d'un noeud dont la variance est inférieure au seuil passé en 3e argument et découpe toute feuille dont la variance est supérieure au même seuil.
— En choisissant deux images, tracez en fonction de h l'évolution du temps de calcul d'un quadtree en utilisant les 3 fonctions ci dessus. On utilisera gnuplot. Comparez ces temps à ceux de la fonction de base `quadtree_split_image(image,double)` utilisant la même valeur de seuil.

3.4 Extension à la couleur

1. Étendre la fonction `split_image` au traitement d'images couleur. On pourra prendre comme critère d'homogénéité la moyenne des variances définie par :

$$\overline{var} = \frac{1}{3M0} \sum_{i=0}^{2} M2[i] - \frac{M1[i] * M1[i]}{M0}$$

2. Testez d'autres critères d'arrêts que la variance pour la découpe du quadtree (facultatif).

4 Remarques

(a) L'objet du rapport n'est pas de dupliquer les commentaires (qui doivent être présent dans le code) mais d'expliquer les différents choix possibles pour chaque problème et les avantages, inconvénients de la méthode retenue.

(b) L'originalité et l'efficacité des solutions algorithmiques (montrées dans le rapport) ainsi que la qualité du code seront des éléments déterminant de la note. Chaque binôme devra faire une démonstration de son projet.

(c) Des images seront disponibles sous /home/public/PROJET_FONDEMENTS