Travaux Encadrés de Recherche

Un algorithme efficace pour le calcul de l'arbre des formes

Enseignant tuteur :
Benoit NAEGEL - Equipe MIV - Laboratoire ICube (Bureau 230)

Objectif :
Implémenter en C++ l'algorithme présent dans l'article "*A quasi-linear algorithm to compute the tree of shapes of n-D images*" en utilisant la librairie *lib-Tim* (développée par l'équipe MIV)

OUHMICH Farid - 28 février 2015

Rapport intermédiaire

Ce rapport est une partie du rapport final, il contient une courte introduction sur les arbres des formes, présentant la problématique ainsi que la démarche suivie pour arriver à la solution souhaitée.
Il évoque également certains des exemples qui seront déroulés proprement par la suite pour aider à la compréhension.

Table des matières

Introduction

Pour commencer, nous reviendrons sur la définition des arbres des formes ainsi que les multiples applications qu'ils peuvent avoir dans les traitements d'images. Nous poursuivrons en explicitant les techniques actuelles de calcul de ces arbres et en évoquant leurs limites. Ensuite, nous montrerons comment l'algorithme que l'on souhaite mettre en place tente de parer ces difficultés. Enfin, plusieurs tests seront effectués sur celui-ci pour mettre en exergue son efficacité.

Arbre des formes

L'arbre des formes (*tree of shapes*) d'une image est une structure permettant d'avoir la représentation d'une image à partir de ses lignes de niveaux. On part de l'hypothèse qu'un pixel sait s'il est plus clair, aussi clair ou plus sombre que ses voisins, et que cette information peut être propagée. Pour récupérer cette information, on utilise les **lignes de niveaux**.
Voici l'exemple d'une image avec ses lignes de niveaux :

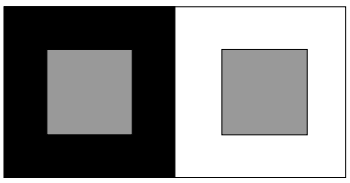

FIGURE 1 – image de base en niveau de gris

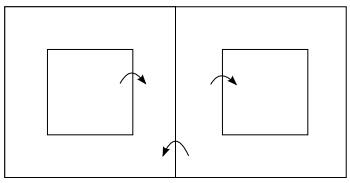

FIGURE 2 – image avec lignes de niveaux

Les flèches signifient "*plus clair que*"

Une des propriétés intéressantes de ces lignes de niveaux et qu'elles sont invariantes aux changements de contraste.
Une autre utilité de ces arbres et de pouvoir obtenir une séparation hiérarchique (au sens de l'inclusion) des différentes régions d'une image, ce qui permet une efficacité accrue pour les calculs d'histogrammes ou de moyennes sur des régions spécifiques.
Les arbres des formes peuvent être utilisés sur des images pour l'utilisation de différentes applications comme la **compression**, la **détection de bords**, la **reconnaissance de formes**, ou encore l'**analyse de texture**.

Techniques actuelles de calcul d'arbres des formes

La méthode utilisée par les algorithmes les plus performants actuellement consiste à construire deux arbres à partir d'une image donnée, le *max-tree* et le *min-tree*, et les fusionner pour obtient le tree of shapes.

Max et Min-tree

Pour commencer, il faut définir les notions de coupures hautes (*upper cuts*) et coupures basses (*lower cuts*) à partir d'une image en niveau de gris. Pour une valeur λ donnée :

- la coupure haute permettra de récupérer tous les pixels ayant un niveau de gris supérieur ou égal à λ.
- la **coupure basse** permettra de récupérer tous les pixels ayant un niveau de gris inférieur strict à λ.

On peut alors déduire l'ensemble $\mathcal{T}_<(u)$ (resp. $\mathcal{T}_\geq(u)$) qui sera constitué des composantes connexes relativement à la haute (resp. basse) coupure. [1]

Les éléments de ces 2 ensembles donnent naissance à 2 arbres qui seront le *max-tree* pour $\mathcal{T}_<(u)$ et le *min-tree* pour $\mathcal{T}_\geq(u)$

1. L'ensemble $\mathcal{T}_<(u)$ utilisera la $2n$-connexité alors que l'ensemble $\mathcal{T}_\geq(u)$ utilisera lui la connexité duale (la $3^n - 1$-connexité).

Exemple de construction max et min-tree

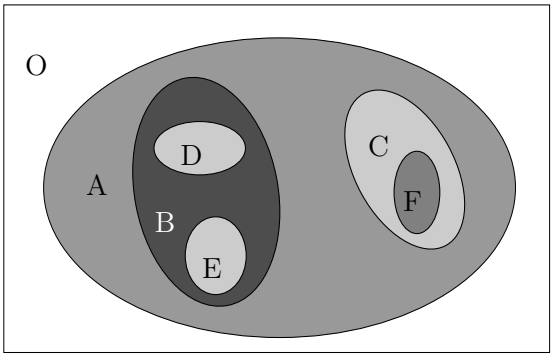

FIGURE 3 – image créée à partir de la *figure 1* de l'article "A quasi linear algorithm to compute tree of shapes of n-D images". On y distingue **4** niveaux de gris différents

Commençons par le max-tree : pour le construire, on doit utiliser les coupures hautes, donc on commencera par les niveaux de gris du plus élevé au plus faible...

État des lieux

Pour le moment voici les différentes parties qui ont étés implémentées et qui fonctionnent correctement :

- Création des *max* et *min-tree* à partir d'une image [2]
- Implémentation de la procédure **Union-Find** permettant justement d'obtenir, à partir d'un tableau trié suivant un certain critère (comme le *niveau de gris* par exexmple), la relation de parenté entre les différents éléments du tableau
- Transformation d'une simple image en sa représentation suivant la *grille de Khalimsky* [3]

Ce qui reste à faire :

- Implémentation de la procédure *Sort* ainsi que la création d'une liste de priorité qui permettront de trier les éléments de la *grille de Khalimsky* pour obtenir en sortie un parcours de l'arbre des formes
- Génération de **tests** sur diverses images pour vérifier que l'algorithme donne bien le résultat attendu

2. Pour le moment la relation de parenté est représentée sous forme d'un simple tableau, un des objectifs sera de pouvoir représenter cette relation sous forme d'un arbre (utilisation de *graphviz*)

3. Cette représentation est une des clés permettant d'obtenir l'arbre des formes sans devoir créer au préalable 2 arbres