Travaux Encadrés de Recherche

Rapport d'erreurs

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

Objectif :
Présenter les erreurs rencontrées lors de l'implémentation de l'algorithme présent dans l'article "*A quasi-linear algorithm to compute the tree of shapes of n-D images*"

OUHMICH Farid

Table des matières

1 Présentation du problème **2**

2 Exemple d'utilisation de la procédure UNION FIND **3**

3 Calcul de l'arbre des formes **4**
3.1 En utilsant les directives de l'article 4
3.2 Version modifiée 6

4 Autres problèmes recontrés **8**

1 Présentation du problème

Ce rapport a été concu pour montrer qu'en appliquant l'algorithme de base [1] sur l'image ci-dessous, on ne réussi pas à obtenir le résultat souhaité.

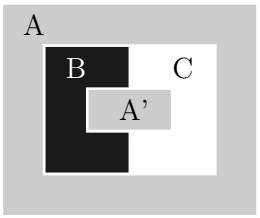

FIGURE 1 – image issue de la figure 3 de l'article de base

1	1	1	1	1	1
1	0	0	3	3	1
1	0	1	1	3	1
1	0	0	3	3	1
1	1	1	1	1	1

FIGURE 2 – la même image avec les niveaux de gris de chaque pixel

Pour rappel, l'ensemble du code est disponible sur le dépôt git, à l'adresse suivante

`https://github.com/farid67/TreeOfShapes.git`

1. Algorithme expliqué dans l'article *A quasi linear algorithm to compute the tree of shapes of n-D images*

2 Exemple d'utilisation de la procédure UNION FIND

Pour commencer, nous allons essayer la procédure UNION FIND sur l'image présentée plus haut pour obtenir le **min-tree**.

Pour calculer le min-tree, on utilise la procédure *computeMinTree*, le résultat obtenu est l'arbre suivant :

```
Arbre : MinTree
Node_0 racine du noeud à l'adresse 9     Nombre d'éléments : 5
         Famille :
                 Père de : Node_1
                 Fils de personne
Node_1 racine du noeud à l'adresse 0     Nombre d'éléments : 20
         Famille :
                 Père de : Node_2
                 Fils de Node_0
Node_2 racine du noeud à l'adresse 7     Nombre d'éléments : 5
         Famille :
                 Père de : personne
                 Fils de Node_1
```

FIGURE 3 – Version texte du min-tree

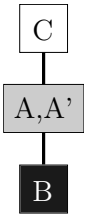

FIGURE 4 – Version schématique du min-tree

On peut également observer les différentes composantes connexes telles qu'elles sont définies par le min-tree en utilisant la procédure *afficheTree*

3 Calcul de l'arbre des formes

3.1 En utilsant les directives de l'article

Avant de se lancer dans la réalisation de cet algorithme, il est nécessaire d'ajouter une bordure externe à l'image dont on souhaite connaitre le TOS (Nous reviendrons sur l'ajout de cette bordure externe dans la dernière section)

Voici donc l'image de base telle qu'elle doit être vue :

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	0	0	3	3	1	1
1	1	0	1	1	3	1	1
1	1	0	0	3	3	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

FIGURE 5 – Image de base

Selon l'article, le cheminement à suivre devrait être le suivant :

Algorithm 1 Algorithm to compute tree of shapes

procedure COMPUTE_TREE_OF_SHAPES(u)
  $u \leftarrow$INTERPOLATE(u)
  $(R, u^b) \leftarrow$ SORT(U)
  $parent \leftarrow$ UNION_FIND(R)
CANONIZE_TREE(u^b,R,$parent$)
return UN-INTERPOLATE(R,$parent$)
end procedure

pour obtenir le résultat suivant :

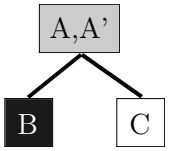

FIGURE 6 – Arbre des formes que l'on devrait obtenir

Nous ne détaillerons pas la première étape, celle de l'interpolation, car on constate visuellement que les résultats sont corrects et que l'image qu'on obtient en retour est bien celle attendue.

Ce qui nous interesse ici, c'est plutôt la procédure SORT qui prend donc en paramètre l'image précedemment interpolée et que l'on considère pour la suite correcte.

La procédure Sort renvoie :

— une image u^b qui servira pour la gestion de la liste de priorité, ainsi que pour la procédure CANONIZE_TREE, on dira grossièrement que c'est la représentation de l'image qu'on obtient en effectuant la procédure INTERPOLATE mais en ne gardant qu'une valeur par élément [2]

— un tableau r dont la lecture correspond à un parcours en profondeur de l'arbre des formes

On effectue ensuite la procédure de canonization, et on observe bien 3 composantes connexes [3] dans le résultat

Le problème se pose lorsqu'on effectue la désinterpolation sur le tableau *parent*, normalement chaque élément de ce tableau devrait avoir une correspondance dans l'image de base pour cela, on utilise un "table de correspondance", on sait par exemple que l'offset 32 dans le tableau parent correspond à l'offset 0 dans l'image de base (c'est donc le premier pixel de l'image= il y a une correspondance pour la valeur 288 qui est, comme on pouvait s'y attendre l'offset du premier pixel de la composante B (cf Figure 1)
Par contre la valeur 232 ne correspond à rien dans l'image de base, ce qui signifie que cette case est issue de la phase d'interpolation et qu'elle est totalement fictive si on se place dans l'image de base.

Le résultat obtenu est le suivant :

2. Dans l'image interpolée, les éléments dits *D-0* ou *D-1* peuvent prendre toutes les valeurs comprises dans un interval

3. le tableau ne contient que 3 valeurs : 32, 288 et 232

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	18	-1	-1	0	0
0	0	18	0	0	-1	0	0
0	0	18	18	-1	-1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

FIGURE 7 – tableau parent obtenu, les cases à -1 sont issus d'une "mauvaise" désinterpolation

3.2 Version modifiée

En partant de l'algorithme de l'article, nous avons essayé de trouver une solution pour que tous les éléments contenus dans le tableau parent avant la phase de désinterpolation correspondent à "quelque chose" dans l'image de base.
On peut facilement se rendre compte que le problème ne vient pas de la procédure INTERPOLATE qui donne un résultat correct. C'est également le cas pour la procédure SORT car le parcours de r correspond bien à ce que nous souhaitons obtenir dans l'arbre des formes. Pour corriger le problème précedemment rencontré, nous nous sommes donc basé sur le retour de la procédure SORT.

On commence donc par supprimer tous les éléments du tableau r qui sont uniquement issus de l'interpolation.

C'est ensuite à partir de ce tableau (qu'on appelera r_clean) qu'on appliquera la procédure UNION FIND.

Il ne sera donc plus nécessaire d'effectuer de "désinterpolation" sur le tableau *parent* parce que ce dernier contiendra autant d'éléments qu'il n'y en a dans r.

Le résultat qu'on obtient en ayant appliqué la procédure décrite ci-dessus est l'abre suivant :

Contrairement à la version précédente, on réussi à obtenir un arbre, cependant ce dernier n'est toujours pas correct, en effet normalement la composante AA' devrait avoir comme fils les composantes B et C.

La conclusion de cette analyse mène à 3 principales hypothèses :

```
Arbre : Tree of Shapes
Node_6 racine du noeud à l'adresse 0     Nombre d'éléments : 46
         Famille :
                 Père de : Node_7
                 Fils de personne
Node_7 racine du noeud à l'adresse 18    Nombre d'éléments : 5
         Famille :
                 Père de : Node_8
                 Fils de Node_6
Node_8 racine du noeud à l'adresse 20    Nombre d'éléments : 5
         Famille :
                 Père de : personne
                 Fils de Node_7
```

FIGURE 8 – Version texte du tree of shapes

FIGURE 9 – Version schématique du tree of shapes

— L'arbre présenté dans l'article n'est pas celui que l'on doit obtenir
— le code comporte des erreurs (possible mais peu probable vu le nombre de vérifications et la quasi-similarité avec ce qui est indiqué dans l'article)
— l'algorithme à utiliser n'est pas complet, il manque une étape entre les procédures SORT et UNION FIND qui semblent être toutes les 2 correctes.

4 Autres problèmes recontrés

Ici nous allons rapidement évoquer les autres problèmes rencontrés (ou qui pourraient apparaître en essayant l'algorithme sur d'autres images).

L'un des principaux problèmes qui pourrait être rencontré le serait à cause de l'ajout de la bordure externe.
Selon l'algorithme, cette bordure externe doit être composée d'une unique valeur étant la médiane des valeurs de la bordure interne (dans notre cas c'est assez simple, il n'y a que des 1) mais prenons par exemple l'image suivante :

0	0	3	3
0	1	1	3
0	0	3	3

FIGURE 10 – Image test pour expliquer le problème de l'ajout d'une bordure externe

Dans ce cas, la bordure interne est composée de 10 valeurs (cinq 0 et cinq 3), la médiane est donc de 1 si on arrondi à l'entier inférieur [4]. Lorsqu'on effectue le calcul du tree of shapes, on aura la composante connexe constituée de cette bordure externe ainsi que des deux pixels centraux qui sera le père des deux autres composantes, or ce résultat n'est pas correct !

Ce qui semble avoir été fait pour la *figure. 5* de l'article de base et qui serait plus logique, serait de prendre la valeur du coin supérieur gauche de l'image et de construire la bordure externe à partir de cette valeur.

À noter aussi que nous avions considérer que le tableau obtenu en retour de la procédure SORT était correct, mais il est possible qu'il soit faux, en effet, l'arbre obtenu dans la partie 3.2 correspond bien à un parcours en profondeur de l'arbre que l'on souhaite avoir (cf Figure 6) , mais le parcours de l'arbre que l'on obtient est le même (cf Figure 9). Cependant en modifiant la valeur des niveaux de gris de l'image mais en gardant toujours un interval suffisant entre les composantes B et C pour faire communiquer A et A' (par exemple en donnant aux pixels de A et A' la valeur 7, aux pixels de B la valeur 5 et aux pixels de C la valeur 10) on obtient un arbre avec C comme parent de B. À ce moment là, le parcours de R ne colle plus, c'est la raison pour laquelle on a considéré que le problème ne venait pas de la procédure SORT.

4. le problème serait le même avec un arrondi sur l'entier supérieur et en ayant des 2 pour les pixels centraux de l'image