Segmentation de maillages 3D surfaciques

Hadrien Croubois
UCBL Lyon 1 – ENS de Lyon
M2 IGI

hadrien.croubois@ens-lyon.fr

Hélène Perrier
UCBL Lyon 1
M2 IGI

perrier.helene@univ-lyon1.fr

CONTENTS

***Abstract*—Dans le cadre de ce TP, nous avons eu l'occasion de développer un algorithme de region-growing pour faire de la segmentation de maillages 3D basée sur sur une mesure d'angles. Nous utilisons la librairie CGAL pour utiliser le maillage comme une carte combinatoire.**

I. MÉTHODE

Le region-growing est une méthode de segmentation basée régions. Elle fonction de la manière suivante.

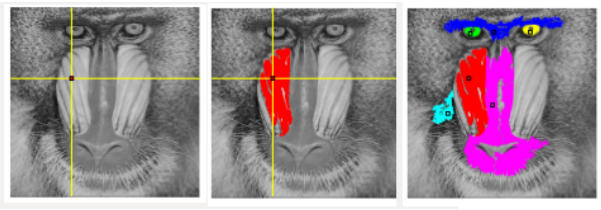

Fig. 1. Exemple de Region Growing sur une Image

On choisis des « seed » aléatoirement dans l'objet à segmenter (qui peut être une image un maillage ou tout autre chose) et l'on fait croître les régions autour en testant certaines caractéristiques. Si elles sont suffisamment similaire, on agrandit la région, sinon on s'arrête.

Parmi les limitations de cette méthode vient le choix des « seeds ». En effet, si on les choisit toute dans la même région, on sera obligé d'exécuter l'algorithme plusieurs fois en recalculant des seeds parmi les éléments hors de toute région pour obtenir la totalité des régions du maillage.

Ici, nous avons implémenté un algorithme de region-growing sur des maillages surfaciques 3D. Pour ce faire, nous utilisons les différences d'angles entre les faces, basées sur les produits scalaires de leurs normales.

Pour contourner la limitation sur le choix des seeds, nous ne partons pas de faces choisies aléatoirement. En effet, il est extrêmement rapide d'initialiser une face en tant que région aussi nous avons initialiser toutes les faces du maillages comme régions. Nous faisons ensuite grandir ces régions les unes après les autres en les fusionnant (avec un algorithme de type UnionFind, détaillé dans la partie Implémentation) et en marquant les régions terminées pour ne pas y revenir. Cela permet de s'assurer d'avoir toutes les régions possibles de l'objet en une seule passe. Ce choix vient aussi de la difficulté dans le cadre d'une carte combinatoire d'accès à une face au hasard ; on retrouve la difficulté des listes chaînées, on doit parcourir tous les brins précédents pour accéder à cette face. Cela impose donc des parcours de la carte systématiques donc on ne perds pas vraiment de temps en la parcourant une fois en entier à l'initialisation.

Dans cette implémentation, nous partons donc de la première face du maillage que nous comparons avec les faces voisines (qui partagent une arête avec la face courante) et suivant si le critère d'arrêt est atteint ou non on continue à tester. L'efficacité de l'algorithme va ainsi entièrement dépendre du choix de ce critère.

II. CRITÈRE D'ARRÊT

Au cours de notre travail nous avons pu étudier différents critères d'arrêt ainsi que leurs impact sur la segmentation.

A. Critère Local

Le premier critère que nous avons implémenté à été un critère complètement local. Nous comparons la face courante avec un de ses voisines et si l'angle entre les deux (obtenu par produit scalaire) est supérieur à un seuil, on fusionne les faces, sinon l'arête est considérée comme vive et devient une frontière de la région.

On obtient les résultats Figure 2 et 3.

On constate que sur des maillages tels que le Stanford Bunny, la segmentation ne se fait pas du tout. En effet, le critère choisi est purement local et ne permet de segmenter que sur des arêtes vives. Il fonctionnera à merveille sur des objets types CAO ou industriels mais sur des maillages d'objets naturels (relativement lisses) on va se retrouver bloqués. Cela va également être un problème pour segmenter des objets au maillage très fin ou adaptatifs, en effet, les faces seront toutes

Fig. 2. Résultats de la segmentation. Régions après l'initialisation – seeds – (à gauche) et l'objet segmenté avec une condition telle que le produit scalaire des normales soit inférieur à 0.5 (à droite)

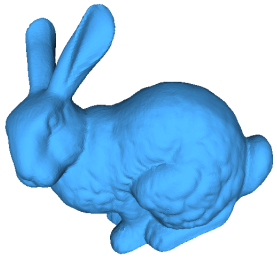

Fig. 3. Segmentation avec un seuillage de 0.7 sur le stanford bunny. On constate que tout le maillage n'est qu'une seule région

très proches les unes de autres et sans arêtes vives elles ne changeront pas radicalement de direction.

B. Critère Global

Nous avons donc laissé de coté le critère local pour nous intéresser à un critère de segmentation global. Ici, au lieu de comparer la face courant avec sa voisine, on conserve dans chaque région la moyenne de toutes les faces rencontrées et on fait le test entre la normale à la face voisine et cette moyenne.

Cela pose un problème au niveau des bordures, en effet, la segmentation devient dépendante de l'ordre dans lequel on choisit les points. Sur des maillages comme l'Armadillo ou le Stanford Bunny, on a tellement de faces avec peu de différence de l'une à l'autre, aussi la variabilité ne sera pas visible (mais elle sera présente néanmoins). En revanche, sur des maillages type CAO, cela peut devenir très problématique.

On observe aussi un autre phénomène, la dérive de la normale de la région. En effet, au fur et à mesure que l'on ajoute des éléments, la moyenne dérive et on peut retrouver le problème du critère local, le test effectué aux bordures de la zone peut n'avoir plus rien à voir avec celui effectué à l'entrée de la zone.

On obtient les résultats présentés Figure 4 et 5

C. Normale Fixe

Pour pallier à ce problème de dérive des caractéristiques de la zone, on a testé un troisième critère. Dans ce cas là, on garde en permanence le même critère que celui d'entrée de la zone, on fait le test par rapport à la normale de la face « seed » de la zone.

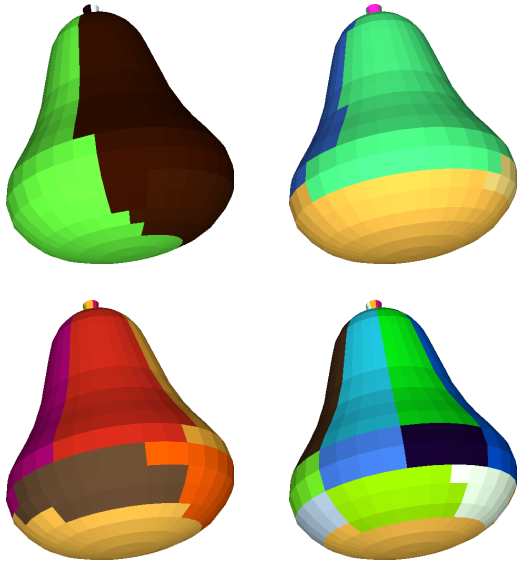

Fig. 4. Segmentation global appliqué à un modèle pressentant peu d'arêtes vives. Les thresholds utilisés sont : 0, 0.5, 0.8 et 0.95

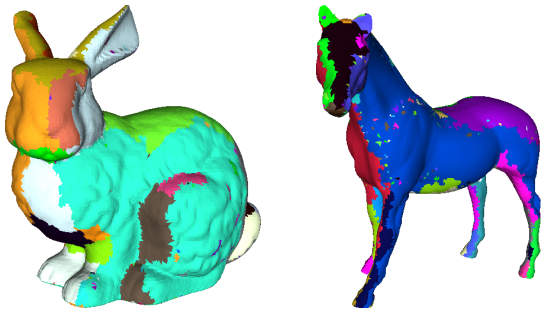

Fig. 5. Application de la méthode de segmentation globale à des gros modèles

Le problème de cette méthode devient donc de le choix des « seed », déterminant pour l'efficacité de l'algorithme. Sur des objets de type sphère, où toutes les faces contiennent la même information, tout se passera bien mais sur des maillages issus de scans, possiblement bruités, on ne peut plus prendre les « seeds » au hasard.

III. Extensions

Parmi les extensions possible du region-growing, on pourrait faire grandir toutes les régions en parallèle ce qui permettrait d'appliquer un algorithme type ligne de partage des eaux. Pour ce faire, il suffirai de définir une fonction de hauteur sur le maillage (dépendent par exemple de la courbure).

Si on ne s?intéresse plus au region-growing mais uniquement à la segmentation, on pourrait aussi exploiter la squelettisation du maillage pour séparer les zones en fonction de la taille de leur boule maximum. Cela reviendrait à exploiter des propriétés de courbure de maillage .

Cela peut aussi se faire en plaçant le critère d?arrêt sur la

valeur de courbure des zones du maillage au lieu de le faire sur les angles entre les faces. On pourrait ainsi séparer les ailes d'un avion ou les jambes du corps d'un animal. Cela permettrait un maillage peut être plus sémantique de l'objet.

IV. Implémentation

A. Organisation du code

...

B. UnionFind

Cet algorithme très simple permet d'organiser des ensembles d'objets en clusters. Ici, les régions.

Pour ce faire, on utilise un représentant pour chaque région associé à l'identifiant de la région. Ensuite, à chaque fusion de région, on fait pointer le représentant de la deuxième région sur celui de la première région, cela à pour effet de mettre tous les éléments de la deuxième région dans la première.

Cette implémentation diffère de l'UnionFind classique dans la mesure où l'on ne peut pas distinguer les éléments contenus dans une région, mais ici via les attributs de CGAL, la seule information dont on a besoin c'est de récupérer la région depuis un élément.

V. Temps d?exécution