HAL open science

Reconstitution de surface non-convexe par algorithme récursif

Claire Guilbaud, Annie Luciani

To cite this version:

Claire Guilbaud, Annie Luciani. Reconstitution de surface non-convexe par algorithme récursif. Journées courbes, surfaces et algorithmes, 1999, pp.1-9. hal-00867520

HAL Id: hal-00867520

https://hal.science/hal-00867520v1

Submitted on 24 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

470

Reconstruction de surface non-convexe par algorithme récursif

Claire Guilbaud
Annie Luciani
ACROE, INPG, 46 avenue Félix Viallet, 38031 Grenoble Cedex
cguilbaud@ifrance.com
Annie.Luciani@imag.fr

Résumé :

Nous présentons une nouvelle méthode récursive pour reconstruire une surface non-convexe à partir d'un ensemble de points dispersés dans l'espace. Ces points sont issus de la modélisation d'un objet obtenu par simulation physique particulaire. La méthode récursive se déroule dans l'espace discret et a pour principe la construction de voxels pour joindre les différents points du modèle physique sous certaines conditions. La surface ainsi construite englobe au mieux les points du modèle physique. Nous utiliserons l'algorithme des Marching Cubes pour effectuer le rendu.

Mots-clés :

modélisation physique, reconstruction de surface, enveloppe non-convexe, visualisation réaliste 3D de formes complexes, espace discret.

Introduction :

Dans les domaines de l'image de synthèse, de nombreux travaux ont été réalisés pour modéliser des objets déformables complexes tels que les phénomènes naturels et turbulents (sable, fluide, fumée). Pour modéliser de tels objets, l'ACROE a choisi d'utiliser le modèle physique dans lequel chaque objet est une structure soumise à des actions basées sur le principe de la physique du point de Newton. Ce principe de modélisation physique particulaire ne permet pas de visualiser des phénomènes complexes dans la mesure où les données issues d'une simulation physique sont représentées sous la forme d'un ensemble de points répartis dans l'espace. Nous nous intéresserons plus particulièrement à la reconstruction de surface de modèles de pâte dentifrice.

Il a donc fallu chercher une méthode permettant de visualiser ces données en créant une impression de volume. La section 1 présentera succinctement la modélisation physique particulaire à l'aide de Cordis-Anima. Nous expliquerons dans la section 2 pourquoi nous n'avons pas utilisé les méthodes classiques de modélisation. La section 3 présentera le principe de l'algorithme. Le rendu sera explicité en section 4. Enfin, nous concluerons et parlerons des travaux futurs en section 5.

1 - La modélisation physique particulaire

1.1 - Cordis-Anima

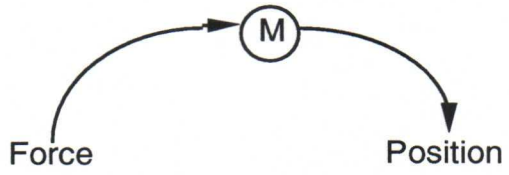
Pour modéliser des objets nous avons utilisé le formalisme Cordis-Anima.

Cordis-Anima est un système de modélisation et de simulation numérique d'objets physiques visibles, audibles et manipulables. Il s'agit d'un langage de programmation. Un

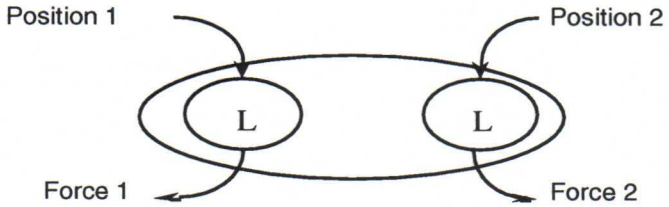
objet physique est une structure soumise à des actions basées sur les principes de la mécanique du points de Newton.

Un objet Cordis-Anima est un ensemble de sous-objets entre lesquels sont établis des liens (connexions de points de communication). Ces sous-objets sont formés de plus petits éléments.

- **l'atome <MAT>** : le point matériel



- **l'atome <LIA>** : l'élément de liaison



- **les atomes spécifiques** : par exemple le sol qui ignore les entrées et retourne toujours la même position

Pour créer un objet Cordis-Anima, il faut définir les paramètres initiaux du système physique, et décrire les évolutions du modèle au cours du temps en effectuant un échantillonnage temporel.

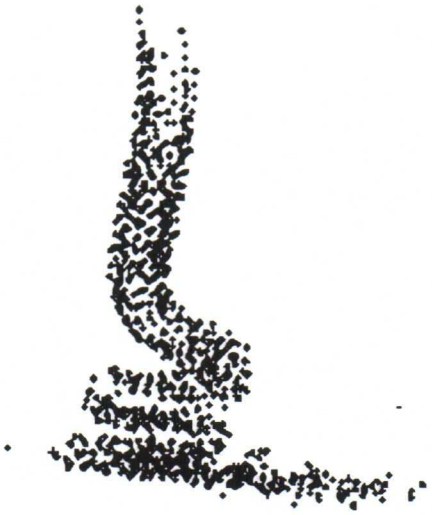
1.2 - Un exemple de simualtion de pâte 3D à l'aide de Cordis-Anima

Nous présenterons brièvement dans cette sous-section un modèle de pâte de type de dentifrice réalisé. Cette présentation est faite pour que le lecteur comprenne quelles sont les données d'entrées de l'algorithme de reconstruction de surface non-convexe, et surtout l'utilité d'un tel algorithme pour visualiser des modèles physiques de fluides.

La pâte a été représentée par un ensemble de masses ponctuelles reliées par une ou plusieurs interactions de types viscoélastiques.

La matière ainsi obtenue a été placée dans un entonnoir et s'écoule dans une boîte.

La simulation obtenue est satisfaisante : le comportement est bien celui du dentifrice sortant du tube que l'on laisserait couler sur une table. Il a été constaté que modéliser une pâte à partir d'un modèle physique ne nécessite pas la prise en compte de nombreuses molécules constituant en temps normal la pâte. Pour réaliser une telle simulation, nous n'avons utilisé que 900 points matériels pour représenter la matière.



Au vue de cette image, nous distinguons bien un volume. Néanmoins il est impossible de discerner la surface de la pâte car celle-ci est inconnue. On ne peut donc lui appliquer de texture ou même la représenter par un simple cylindre généralisé de couleur uniforme.

Pour obtenir une pâte qui ait toutes les caractéristiques visuelles du dentifrice, il faut la visualiser de manière plus précise. Nous allons donc chercher à implémenter une méthode de visualisation de formes complexes.

2 - L'utilisation des méthodes classiques

Nous cherchons à réaliser une visualisation "semi-automatique" des différentes parties constituants un modèle physique. Par semi-automatique, il faut comprendre permettre à l'utilisateur de choisir la méthode de rendu, et donc la forme à assigner à une partie d'un modèle physique. Cette visualisation doit être le choix de l'utilisateur car ce dernier doit rester maître des formes à donner au modèle. Par exemple, lorsque l'objet physique est un drapeau, le mât pourra être représenté par une surface implicite dont le squelette de base est une ligne, tandis que le fanion le sera par une surface implicite avec un squelette de base de plan.

Comme nous voulions visualiser un nuage de points sans structure prédéfinie (cf. section 1), nous avons essayé d'utiliser une des méthodes déjà existante qui permet de modéliser, puis visualiser des formes volumiques.

La *CSG* n'est pas adaptée car il est pratiquement impossible de choisir les primitives de base en fonction de la position des points. Nous avons ensuite pensé à utiliser des *surfaces implicites* car ces dernières permettent d'animer des flux de liquides. De plus leur utilisation fournit des surfaces lisses. Les blobs sont bien adaptés pour trouver un volume englobant qui tienne compte des interactions entre particules du modèle. Blinn a démontré ([Blin82]) la valeur de cette technique à la fois pour la modélisation et pour la visualisation. Le problème qui se pose pour le cas qui nous intéresse, est la façon d'attribuer des valeurs aux différents paramètres des blobs. Quel seuil utiliser ? Quel rayon pour un blob donné ? Comment positionner les blobs en fonction du nuage de points pour ne pas obtenir un effet "intestin" indésirable ?

L'utilisation de surfaces implicites étant une solution à de nombreuses inconnues, nous en avons préféré une autre : serait-il possible d'utiliser de *NURBS* pour reconstruire le modèle. S'agissant de splines d'approximation la surface se situera à l'intérieur de l'enveloppe convexe des points de contrôle. Un autre problème est le choix de ces points de

contrôle en fonction des points caractéristiques de l'objet modélisé. Mais comment connaître ces points caractéristiques ? Cette méthode n'est donc pas adaptée au cas de la visualisation de modèles physiques.

Une autre approche des surfaces implicites est d'utiliser une distance contrainte i.e. utiliser un *squelette pour définir la surface en terme de distance au squelette* ([Bitt95], [Ferl96]) . Les objets ne possédant pas tous un axe médian, ils ne sont pas tous représentables. De plus , il faut trouver une méthode qui à partir d'un nuage de points extrait un squelette. Cette solution n'est pas entièrement satisfaisante, nous en préférerons donc une autre.

Les méthodes dites de *reconstruction de volume 3D* utilisée en imagerie médicale impliquerait des imprécisions. Les données initiales sont des coupes 2D. Le principe est assez simple : il faut mettre en correspondance deux coupes successives pour reconstruire la surface du volume initial. Pour le cas qui nous intéresse cette méthode n'est pas adaptée car nous ne disposons que de points 3D non "découpables" en tranches sans introduire des imprécisions qui se répercuteraient sur la construction du volume au moment de la mise en correspondance de deux tranches successives.

Aucune des solutions précédemment citées n'étant satisfaisante pour visualiser des modèles de pâtes de type dentifrice, nous allons chercher à implémenter un algorithme qui construira une enveloppe englobante pour un ensemble de points.

3 - Principe de l'algorithme récursif de reconstruction de surface

Comme il a été précisé en introduction, nous cherchons une manière de représenter un volume qui n'est constitué que par un nuage de points. Dans un premier temps nous allons définir les différents termes que l'on utilisera pour expliquer le principe de base de l'algorithme, puis ensuite nous traiterons de sa conception.

3.1 - Le vocabulaire utilisé

Un voxel est la représentation 3D d'un pixel. Chaque voxel est une petite unité de volume. L'algorithme n'utilise que des voxels non-cubiques.

L'assemblage de voxels formant une grille 3D sera appelé *volume englobant l'ensemble des points constituants le modèle* ou plus simplement *volume englobant* . Il s'agit d'une notion analogue à celle des bounding boxes en ray-tracing. Sa taille est choisie par l'utilisateur et correspond à *la taille de discrétisation* de l'objet à visualiser.

L'enveloppe est contenue dans le volume englobant, et est constituée d'un ensemble de voxels formant une frontière entre l'intérieur et l'extérieur du volume construit.

La carte de densités possède la même taille que le volume englobant. Elle est composée de valeurs indiquants combien de points du nuage se trouvent dans une zone d'une taille prédéfinie par l'utilisateur.

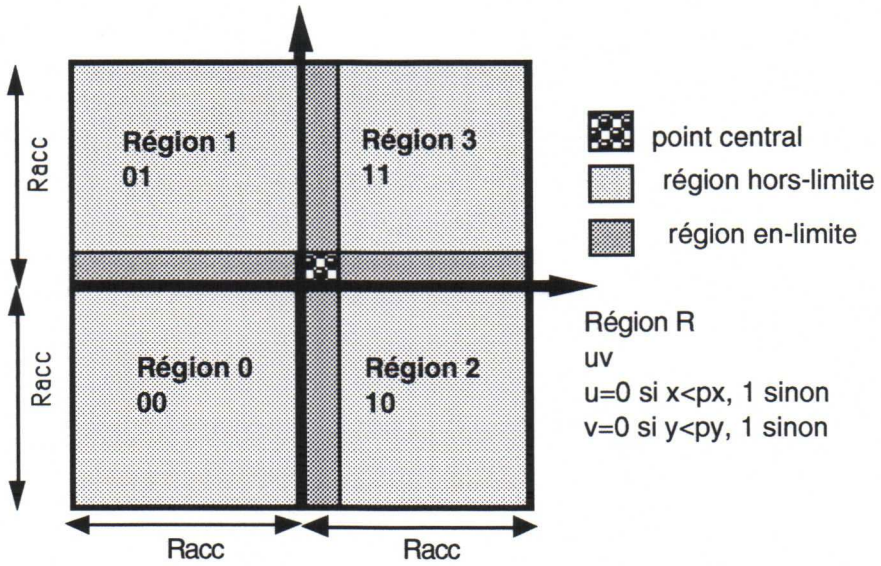
Une zone est un espace clos borné selon les 3 directions classiques. Elle peut être représentée par un pavé (3D) ou un rectangle (2D). Ce terme sera utilisé au cours de la construction de la carte de densités.

Une région est une notion analogue à celle de zone. Un point qui diffère est l'utilisation qui en est faite. On parlera de régions lorsque l'on construira la surface. En fait, une région est un espace plan ou non (2D/3D) qui sont définis autour d'un voxel de coordonnées x_V, y_V, z_V. Par exemple en 2D, il existe 4 régions :

- $x<x_V$ et $y<y_V$;
- $x<x_V$ et $y\geq y_V$;

- $x \geq x_V$ et $y < y_V$;
- $x \geq x_V$ et $y \geq y_V$.

Les régions seront dites réduites si elles sont définies par $x_C < x < x_V$, ...



S'étendre dans une région dans une direction donnée ou vers un voxel signifie créer un chemin 26-connexe entre le voxel courant et le voxel le plus proche de plus forte densité. La création d'un chemin 26-connexe revient à remplir les voxels vides séparant les 2 voxels précédemment cités.

L'algorithme a été conçu pour visualiser de manière réaliste une pâte simulée par modélisation physique.

3.2 - Reconstruction de surface non-convexe par algorithme récursif

L'utilisateur définit 3 paramètres de construction :

- la taille du volume englobant : *discrétisation* ;
- la taille d'une zone : *étage* (nombre d'étages de voisins à prendre en compte pour calculer la carte de densités) ;
- la taille d'une région : *Racc* (taille de la région "raccourcie" dans laquelle on cherchera à s'étendre vers un des voxels le plus dense et le plus proche).

Chacun de ces paramètres tient un rôle important pour obtenir une surface non-convexe. Nous verrons par la suite le rôle de chacun dans la construction de la surface.

L'algorithme se déroule en 3 étapes.

3.2.1 - La discrétisation de l'espace du modèle

La première phase de l'algorithme consiste à créer le volume englobant qui contient tous les points du modèle physique pour lequel on souhaite reconstruire une surface. Par la suite, le volume englobant est considéré comme un volume discret composé d'un ensemble de voxels non uniformes (ce ne sont pas obligatoirement des cubes, cf. dessin juste après la définition du volume englobant).

Une fois ce volume créé, il faut discrétiser l'espace du modèle de manière à pouvoir associer à chaque point initial un voxel du volume englobant. Cette partie de la première étape consiste à placer dans le volume englobant les points du modèle afin de les repérer de manière simple pour calculer la carte de densités.

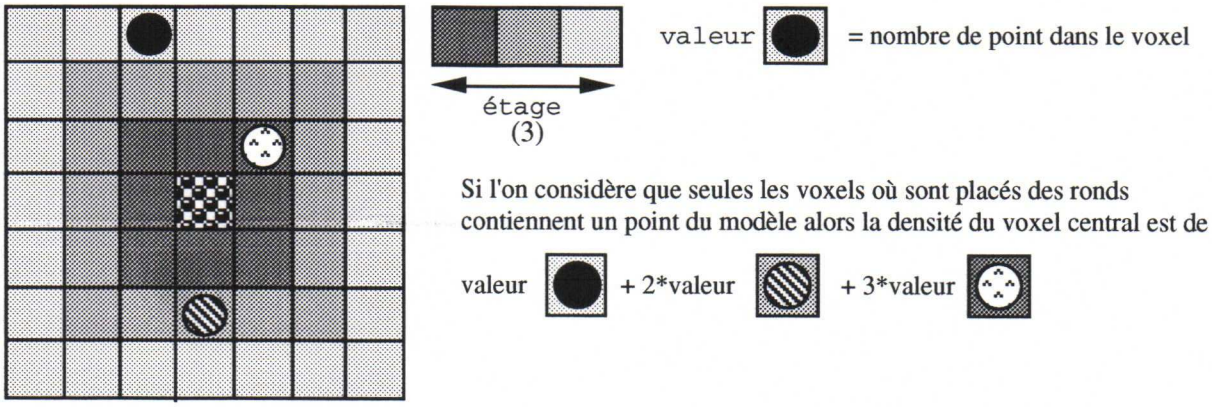
3.2.2 - Le calcul de la carte de densités

Pour calculer la carte de densités, nous n'utiliserons que le volume englobant dans lequel les points du modèle ont été placés.

La taille d'une zone (la variable `étage`) fournit une indication sur l'anisotropie de l'enveloppe à générer en fonction de la discrétisation choisie.

La densité en un voxel représente le nombre de points dans la zone pondéré par une valeur correspondant au nombre de voxels séparant le voxel contenant un point du voxel central. Cette méthode revient à appliquer un masque de convolution sur le volume englobant.

Il existe 2 façons de calculer la carte de densités. Soit on calcule la densité pour chaque voxel du volume englobant, soit on la calcule pour l'ensemble des voxels inclus dans chaque zone de taille `étage` autour des points formant le nuage. Pour une discrétisation de 110, un nuage de points représentant une pâte, une zone de taille 3, et une région de taille 2, on n'obtient pas de différences flagrantes dans la mesure où seuls quelques voxels en surface ont été construits dans des endroits peu nombreux : endroits où la non-convexité de la surface est accentuée.



Il est possible de faire subir un post-traitement de la carte de densité. Ce post-traitement est surtout utile lorsque la densité est calculée pour l'ensemble des voxels du volume englobant. Il se déroule en 2 étapes :

- la phase de seuillage sert à éliminer les voxels en limite de pseudo-enveloppe vers lesquels on ne souhaite pas s'étendre car la zone considérée a une densité trop faible.

- la phase dite de rebouchage est utile dans la mesure où la première phase a éliminé des voxels qui remplissaient bien les conditions d'appartenance à la pseudo-enveloppe de par leurs positions par rapport aux voxels pleins.

Il faut noter qu'au vu des résultats des tests effectués, le post-traitement de la carte de densités n'est vraiment utile que lorsque la densité est calculée pour chacun des voxels formants le volume englobant, et quand on cherche à obtenir une enveloppe 2D. Sa nécessité ne semble pas évidente en 3D.

Cette première étape de l'algorithme récursif de reconstruction de surface a une complexité en O(n) où n est la taille de l'espace dicrétisé.

3.2.3 - La reconstruction de la surface

Nous disposons maintenant d'un ensemble de points placés dans un volume englobant ainsi que d'une carte de densités. Ce que nous souhaitons obtenir est un ou plusieurs volumes non-convexes contenant l'ensemble des points du modèle de l'objet à visualiser.

Pour cela, à l'aide de la carte de densités, nous allons créer par récursion des chemins 26-connexes reliant les différents voxels déjà pleins du volume englobant. A la fin de l'algorithme, il sera impossible de distinguer les différents chemins 26-connexes dans la mesure où chaque nouveau voxel créé au cours de la phase d'expansion sera relié à un voxel donné par autre chemin 26-connexes. L'ensemble de ces chemins formeront le ou les volumes cherchés.

Le principe est le suivant :

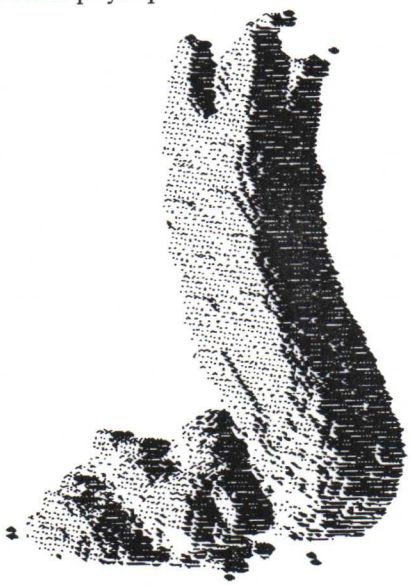
```
Pour chaque point du nuage (indice point_courant)
      voxel_courant  = voxel contenant le point_courant
      Détecte(voxel_courant)

Détecte
Pour chaque région autour de voxel_courant (région_courante)
      Si (Présence d'un ou plusieurs voxels de plus fortes densités
les plus proches dans la région_courante)
      Alors
            Détermination des nouveaux voxels à créer
            Pour chaque nouveau voxel créé (indice nouveau_voxel)
                  Si volume_englobant[nouveau_voxel] est vide
                  Alors
                        volume_englobant[nouveau_voxel]=1
                        Détecte(nouveau_voxel)
```

volume_englobant[i] est le tableau qui contiendra le volume construit. Au début de l'algorithme il s'agit du volume englobant dans lequel les points du nuage ont été placés.

Cette deuxième étape est en O(n).

En représentant les voxels pleins par des sphères, on obtient les résultats suivants :

discrétisation : 110
étage : 3
Racc : 2
aucun post-traitement de la carte de densités
nombre de points du modèles physiques : 900



4 - Rendu du volume construit

Nous disposons maintenant d'un volume discret composé de voxels (cf image à la fin de la section 3).

Les Marching Cubes

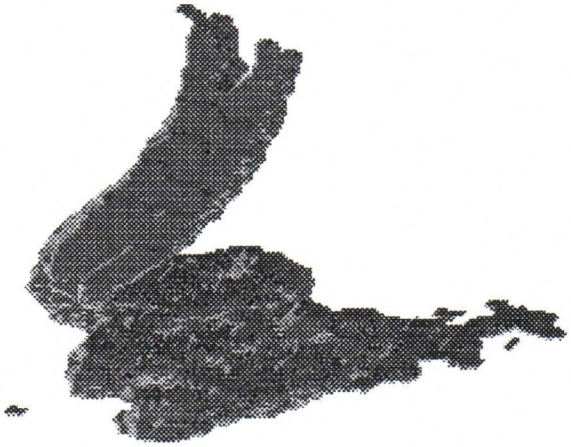
L'algorithme des Marching Cubes (Lorensen et Cline) a été écrit dans le but de visualiser un volume. Il est généralement utilisé pour visualiser une isosurface en construisant des polygones qui l'englobe.

La principale difficulté dans le cas qui nous intéresse est de savoir à quelles conditions un sommet est à l'intérieur ou à l'extérieur du volume à visualiser.

L'algorithme se déroulera comme suit : dans un premier temps, nous générerons les 256 cas généraux à partir des 15 cas de bases, nous calculerons aussi les normales à chaque facette. Puis chaque voxel du volume construit sera identifié à l'un des 256 cas généraux.

Dans le cas qui nous intéresse, nous savons que le volume à visualiser est constitué de voxels, donc la détermination de l'appartenance d'un sommet d'un cube à l'enveloppe finale sera modifiée par rapport à l'utilisation faite de l'algorithme dans les principaux cas.

Différentes études ont été faite pour savoir à quelles consditions un sommet appartient au volume. Il ressort de ces résultats qu'un voxel plein doit être différencié d'un voxel vide, que l'appartenance d'un sommet au volume est fonction du nombre de voisins pleins qui l'entourent. Les voisins d'un voxel non pas tous la même importance dans la détermination de l'appartenance d'un sommet au volume. En effet, un voisin par les faces a plus de poids qu'un voisin par les arêtes ou qu'un voisin par les sommets. La détermination de l'enveloppe au volume revient donc à calculer pour chaque voxel le nombre et le type de voxels voisins pleins.



Cette image représente la surface obtenue après application de l'algorithme des Marching Cubes au volume montré à la fin de la section 3.

5 - Conclusion

L'algorithme a une complexité en O(n), où n est la taille de l'espace discrétisé. La taille de discrétisation influence la construction du volume car plus cette taille est grande, plus il sera possible de visualiser des détails.

Les paramètres choisis par l'utilisateur permettent d'affiner la construction de l'enveloppe, et doivent être représentatifs de la surface voulue. La variable caractérisant la taille de la région restreinte dans la phase d'expansion est importante. En effet si celle-ci est trop grande, nous passerions à côté de détails.

Son principal défaut est son temps de calcul qui est, comme pour la phase de discrétisation du ray-tracing discret, proportionnel à la taille de discrétisation de l'espace. Ce temps est "incompressible " dans la mesure où nous sommes obligés de parcourir le volume englobant presque dans sa totalité.

Néanmoins, ce temps peut être amélioré si nous parallélisons l'algorithme dans sa totalité (calcul de la carte de densités, et construction de l'enveloppe).

Un point intéressant de cet algorithme est la possibilité d'obtenir plusieurs enveloppes. Par exemple, si la pâte se coupe en deux (ou plus), nous pourrons visualiser cette fracture.

Pour de futurs travaux, il serait intéressant de paralléliser l'algorithme récursif de reconstruction de surface non-convexe et de changer de méthode de rendu. En effet l'utilisation des Marching Cubes lorsque la normale est calculée à partir des facettes et non à partir de la surface à visualiser, provoque un "effet Lego" compte tenu du nombre limité d'orientations différentes des facettes générées par la méthode (multiple de 45 degrés).

Bibliographie

[Aman87] J. Amanatides, A. Woo
A Fast Voxel TRaversal Algorithm for Ray-Tracing
Proceedings of Eurographics'87, North-Holland

[Blin82] J.F. Blinn
A Generalization of Algebraic Surface Drawing
ACM Transactions on GRaphics 1,3 (July 1982), 235-256

[Berg92] K.Bergad, A.Atamenia
Approche Discrète du lancer de rayons
2è journées de l'AFIG, Toulouse, décembre 1994

[Bitt95] E. Bittar, N. Tsingos, M.P. Gascuel
Automatic reconstruction of unstructured 3D data : combining medial axis and implicit surfaces
Eurographics'95, Volume 14, septembre 95

[Ferl96] E. Ferley, M.P. Gascuel, D. Attali
Skeletal reconstruction of branching shapes
Implicit Surfaces'96 : 2nd International Workshop on Implicit Surfaces, octobre 1996

[Mura91] S. Muraki
Volumetric shape description of range data using "blobby model"
Computer Graphics, Volume 25, N°4, July 1991

[Yage92] R.Yagel, D.Cohen, A.Kaufman
Discrete Ray tracing
IEEE Computers Graphics and Applications 1992