Description et reconstruction rapide de surfaces.

Laurent CHEVALIER Fabrice JAILLET Atilla BASKURT

29 juin 2000

Table des matières

Chapitre 1

Introduction

Dans de nombreux domaines, il peut être intéressant de disposer de descripteurs performants associés aux surfaces présentes dans une image 3D afin d'obtenir une segmentation et une reconstruction rapide des objets tridimensionnels. Ces descripteurs doivent pouvoir être visualisés en temps réel. Ceci implique une modélisation simplifiée. Une certaine tolérance ou un flou dans la reconstruction de la surface d'un objet permet d'envisager le développement d'un tel modèle (descripteur) réduit.

Le domaine des télécommunications, par exemple, est marqué actuellement par l'explosion des services et applications réseaux multimédia. Ce type d'applications s'avère gourmand en débit et exigeant en qualité, ce qui suppose des avancées à la fois dans le domaine de la qualité et de celui du codage. Une description compacte des données 3D est un avantage indéniable pour effectuer un codage efficace.

Ce rapport présente une méthode de compression géométrique d'objets 3D pour la transmission de scènes réelles et virtuelles. Chaque scène est constituée d'objets géométriques paramétrables. L'objet géométrique élémentaire doit être défini dans une première étape. Il doit permettre la modélisation de surface compacte que l'on peut aisément déformer et assembler. Ce descripteur se doit alors, d'être très généraliste pour assurer compacité et précision à l'approximation et la segmentation de scène 3D (deuxième étape).

Le présent document débutera par un état de l'art du domaine d'application qui nous intéresse en appuyant essentiellement sur les travaux réalisés à partir de superquadriques. Dans une seconde partie, sera présentée la méthode de description de scène mis au point pendant le stage de DEA. Et enfin, une troisième et dernière partie commentera l'implémentation de l'approche et les résultat obtenus.

Chapitre 2

État de l'art

2.1 Superquadriques

2.1.1 Généralités

Les superquadriques sont un modèle de surfaces introduit dans l'informatiques graphique par A.H. Barr en 1981[1]. Elles sont actuellement surtout utilisées en robotique et en vision, étant un descripteur compact et pouvant facilement servir à approximer un objet [2, 3, ?, 4, 5]. Elles sont, de plus, aisément assemblables et déformables[1, 6, 7].

Extension des plus communes quadriques[1], le modèle superquadrique peut être partitionné en 4 sous-classes: les superellisoïdes, les supertoroïdes, les superhyperboloïdes, les superparaboloïdes. Les superellipsoïdes et, dans une moindre mesure, les supertoroïdes sont, en pratique, les seules à être réellement utilisées, les deux autres sous-classes n'étant pas des surfaces fermés. On parlera, par la suite, uniquement des superellipsoïdes mais la plupart des résultats sont en fait généralisables aux superquadriques.

Les superquadriques sont obtenues par le produit sphérique de deux courbes 2D. La superellipsoïde est le produit de deux superellipses et est défini par la formule implicite:

$$\left(\left(\frac{x}{a_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{a_3}\right)^{\frac{2}{\epsilon_1}} = 1 \tag{2.1}$$

avec:

- a_1, a_2, a_3 définissant le facteur d'échelle respectivement sur l'axe des x, y et z.
- ϵ_1 et ϵ_2 définissant la courbure respectivement lattitudinale et longitudinale de la forme.

Les superellipsoïdes vont décrire un éventail très large de formes allant de simples ellipsoïdes ($\epsilon_1 = 1$ et $\epsilon_2 = 1$), aux parallélépipèdes ($\epsilon_1 \to 0$ et $\epsilon_2 \to 0$) en passant par des cylindres ($\epsilon_1 = 1$ et $\epsilon_2 \to 0$) ou des octoèdres($\epsilon_1 = 2$ et $\epsilon_2 = 2$).

1. les quadriques sont équivalentes à la classe des superquadriques telle que $\epsilon_1 = \epsilon_2 = 1$

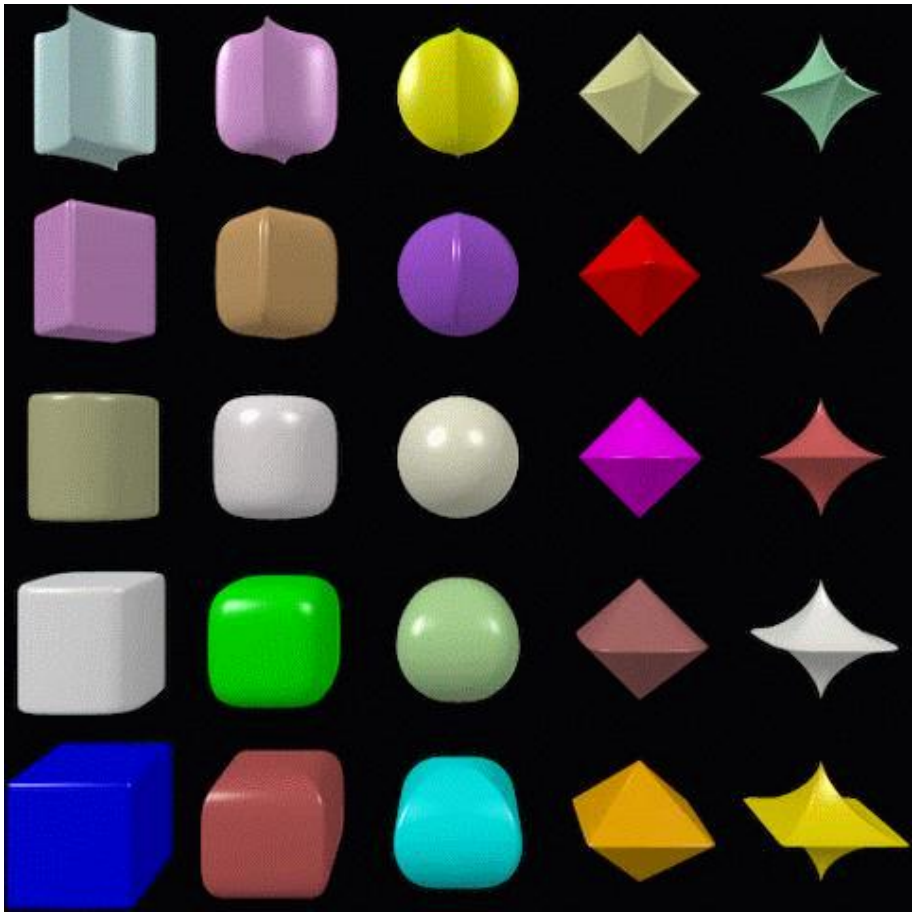

FIG. 2.1 – *Exemple de superellipsoïdes. ($a_1 = a_2 = a_3 =$ constante, ϵ_1 et ϵ_2 variables.)*

Cette équation permet naturellement de définir la fonction intérieur-extérieur:

$$f(x,y,z) = \left(\left(\frac{x}{a_1} \right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{a_2} \right)^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{y}{a_3} \right)^{\frac{2}{\epsilon_1}} \tag{2.2}$$

avec:

- $f(x,y,z) = 1$ si (x,y,z) est sur la surface.
- $f(x,y,z) < 1$ si (x,y,z) est dedans.
- $f(x,y,z) > 1$ si (x,y,z) est dehors.

2.1.2 Paramétrisation

Comme pour les quadriques dont elles dérivent, les superquadriques peuvent aussi être définies par une formule paramétrique. Cette particularité, non vérifiée pour toutes les surfaces implicites, les rendent encore plus appréciable car si les surfaces implicites rendent l'approximation (calcul de distance) ou bien

d'autres problèmes (opération booléennes ...) relativement simples, elle est aussi synonyme d'un affichage assez lourd. La forme paramétrique, quant à elle, permet, entre autre, un affichage très rapide et très simple (l'échantillonnage de points et par là de facettes étant immédiat).

Différentes paramétrisations sont possibles, on pourra les utiliser suivant l'usage. Les exemples [2] suivants ne sont pas exhaustifs.

Paramétrisation standard

La paramétrisation standard des superellipsoïdes découle du produit sphérique de deux superellipses:

$$\begin{pmatrix} x(\eta,\mu) \\ y(\eta,\mu) \\ z(\eta,\mu) \end{pmatrix} = \begin{pmatrix} a_1 \cos^{\epsilon_2}(\mu) \\ a_2 \sin^{\epsilon_2}(\mu) \end{pmatrix} \otimes \begin{pmatrix} \cos^{\epsilon_1}(\eta) \\ a_3 \sin^{\epsilon_1}(\eta) \end{pmatrix} = \begin{pmatrix} a_1 \cos^{\epsilon_1}(\eta)\cos^{\epsilon_2}(\mu) \\ a_2 \cos^{\epsilon_1}(\eta)\sin^{\epsilon_2}(\mu) \\ a_3 \sin^{\epsilon_1}(\eta) \end{pmatrix} \tag{2.3}$$

$\eta \in [-\pi/2, \pi/2], \mu \in [-\pi, \pi]$.

Cette paramètrisation est particulièrement intéressante pour l'affichage, car elle concentre l'échantillonage des points dans les zones de grandes courbures (fig. 2.1.2).

Paramétrisation régulière

Ce second exemple de paramétrisation, à l'inverse du précédent, échantillonne les points de manière uniforme sur l'ensemble de la superellipsoïde (fig.2.1.2).

On l'obtient à partir des équations paramétriques d'une ellipsoïde qu'on multiplie par une fonction ρ et de la formule implicite de la superellipsoïde:

$$\begin{cases} x(\eta,\mu) &= \rho(\eta,\mu) a_1 \cos\eta\cos\mu \\ y(\eta,\mu) &= \rho(\eta,\mu) a_2 \cos\eta\sin\mu \\ z(\eta,\mu) &= \rho(\eta,\mu) a_3 \sin\eta \end{cases} \quad \begin{cases} \eta \in [-\pi/2, \pi/2], \\ \mu \in [-\pi, \pi] \end{cases} \tag{2.4}$$

$$\rho = \left[\left(|\cos\mu\cos\eta|^{\frac{2}{\epsilon_2}} + |\sin\mu\cos\eta|^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + |\sin\eta|^{\frac{2}{\epsilon_1}} \right]^{-\frac{\epsilon_1}{2}} \tag{2.5}$$

2.1.3 Remarque sur les normales

On peut calculer les normales en tout point d'une superellipsoïde grâce aux expressions:

$$\begin{cases} N_x(\eta,\mu) &= \frac{1}{a_1} cos^{2-\epsilon_1}(\eta) cos^{2-\epsilon_2}(\mu) \\ N_y(\eta,\mu) &= \frac{1}{a_2} cos^{2-\epsilon_1}(\eta) sin^{2-\epsilon_2}(\mu) \\ N_z(\eta,\mu) &= \frac{1}{a_3} sin^{2-\epsilon_1}(\eta) \end{cases} \quad \begin{cases} \eta \in [-\pi/2, \pi/2], \\ \mu \in [-\pi, \pi] \end{cases}$$

2. On notera, pour simplifier l'écriture, $cos^a x$ pour $signe(cos(x)) * |cos(x)^a|$ (de même pour $sin^a x$).

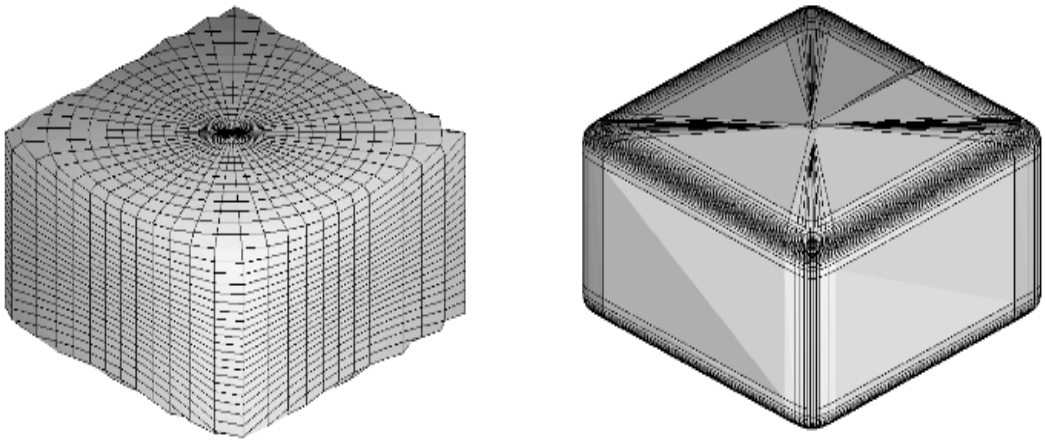

FIG. 2.2 – *Paramétrisation régulière et standard.*

On peut donc remarquer que si ϵ_1 et ϵ_2 sont inférieurs à 2, alors l'équation des normales définit une autre superellipsoïde (sa duale) de paramètre $\epsilon'_1 = 2-\epsilon_1$ et $\epsilon'_2 = 2-\epsilon_2$.

2.2 Approximation d'un nuage de points par une superquadrique

Les données peuvent provenir d'un scanner 3D ou de tout autre capteur, mais se présentent généralement en définitif, comme un nuage de points de R^3 non organisé[3] (c'est à dire sans information de relation entre ces points).

Il faut alors trouver un modèle correspondant aux données [8], c'est à dire déterminer pour quelles valeurs de ses paramètres ce dernier pourra représenter tous les points du nuage. Bien évidemment, ce modèle ne peut généralement pas correspondre exactement aux données et on se limitera donc à trouver la meilleure approximation.

Grâce à l'éventail de formes très large et leur faible nombre de paramètres, les superquadriques sont un modèle très intéressant pour l'approximation. C'est A. Pentland qui le premier les utilisera dans ce domaine.

2.2.1 Approximation au sens des moindres carrés

Solina propose la méthode la plus largement répandue pour approximer des données tridimensionnelles avec une superquadrique. L'approximation se fera au sens des moindres carrés. On dira qu'un n-uplet est meilleur qu'un autre si la somme des distances de chaque point à la superquadrique est plus faible. Il s'agit donc de trouver:

3. On considérera désormais les données comme un ensemble de N points de R^3

$$\min \sum_{i=0}^{N} D(p_i)^2 \tag{2.6}$$

avec $D(p)$ la fonction distance du point p à la superquadrique.

Dans un premier temps, il s'agit donc de déterminer D. À première vue, $f-1$ (fonction intérieur-extérieur à laquelle on retranche 1) semble pouvoir convenir. En effet $|f-1|$ vaut 0 sur la superquadrique et grandit plus on s'en éloigne. En fait, cela n'est pas suffisant. Des points, à même distance euclidienne de la superquadrique, donnent des valeurs différentes avec $f-1$ lorsque ϵ_1 est petit ($\epsilon_1 < 0.01$). Solina a démontré qu'en prenant $F = f^{\frac{\epsilon_1}{2}}$ le problème disparaissait sans changer la forme de la superquadrique (i.e. ce sont les mêmes points p qui vérifient $f(p) = 1$ et $F(p) = 1$). On déterminera donc:

$$\min \sum_{i=0}^{N} (f(p_i)^{\frac{\epsilon_1}{2}} - 1)^2 \tag{2.7}$$

en considérant que $f(p_i)^{\frac{\epsilon_1}{2}}$ est une bonne approximation de la distance de p_i à la surface.

Prise en compte du volume

L'unique considération de la distance des données au modèle n'est pas suffisante surtout lorsque le nuage est ouvert. En effet, plusieurs superquadriques de taille très diverses peuvent approximer le même nuage de points. Le volume de la superquadrique doit être pris en compte dans l'approximation.

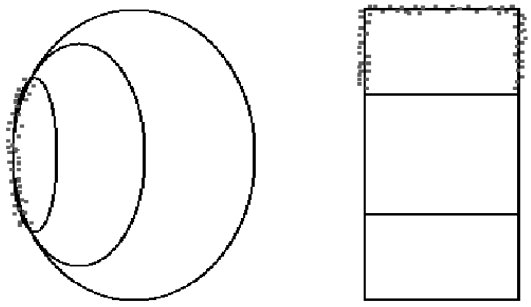

FIG. 2.3 – *Dans ces deux exemples un très grand nombre de superquadriques peuvent approximer le nuage, bien sûr c'est la plus petite qui est la meilleure.*

Cette prise en compte se fait en multipliant 2.7 par une fonction dépendante du volume. Solina a montré, par expérimentation, que pour les facteur du type $(a_1 a_2 a_3)^{\alpha}$, le meilleur choix était l'ajout du facteur $\sqrt{a_1 a_2 a_3}$ devant la fonction de minimisation. Il est important de noter que cet ajout ne crée pas de nouveau minimum. Quand $a_1 = a_2 = a_3 = 0$, la fonction tend vers l'infini.

En définitif, il s'agit, en tenant compte de tous les paramètres de la superquadrique, de la translation (t_i) et de la rotation (r_i)éventuelles, de minimiser la fonction:

$$\sqrt{a_1 a_2 a_3} \sum \left(f\left(x_i, y_i, z_i; a_1, a_2, a_3, \epsilon_1, \epsilon_2, t_x, t_y, t_z, r_x, r_y, r_z\right)^{\frac{\epsilon_1}{2}} - 1 \right)^2 \qquad (2.8)$$

2.2.2 Méthodes de minimisation

L'approximation par une superquadrique est un problème nécessitant un algorithme de régression non-linéaire sur 11 variables.

Dans le cas des quadriques, Cernuschi-Frías approxime grâce à une méthode directe, dérivée de celle Bookstein (cas 2D des coniques). Il trouve le minimum de la somme des erreurs au carré (qui n'est pas la somme standard des différences) en résolvant un problème de vecteurs propres/valeurs propres sur une matrice 6*6.

Dans le cas des superquadriques, la méthode directe n'existe pas. On est, donc, obligé de passer par des méthodes itératives.

Méthodes itératives classiques

Ce sont des algorithmes, originaires de méthodes numériques [**?**], plus que largement utilisés. Ils utilisent principalement les techniques de descente de gradient pour, pas à pas, se rapprocher de la solution.

Dans cette catégorie et pour notre problème, l'algorithme de Levenberg-Marquardt [**?**] fait quasiment l'unanimité. C'est d'ailleurs celui que propose d'utiliser Solina. Cet algorithme converge très rapidement vers une solution et ne nécessite que les dérivées premières (calculables analytiquement).

Le principal défaut de ce type d'algorithme est qu'il ne détermine qu'un minimum local.

Méthodes stochastiques

Pour réellement trouver le minimum global, les méthodes stochastiques introduisent de l'aléatoire pour sortir du piège des cuvettes (minimums locaux). Ces approches sont souvent assez lourdes et donc peu utilisées. Elles sont néanmoins envisageables, on peut citer les algorithmes génétiques ou le recuit simulé.

2.2.3 Première évaluation des paramètres

Avoir une première évaluation précise des paramètres est nécessaire avant même d'appliquer l'algorithme de minimisation. Cela permet, bien sûr, dans tous les cas, d'accélérer la régression. Mais surtout, le résultat dépendra totalement de cette évaluation dans le cas des méthodes ne déterminant qu'un minimum local.

Les paramètres de courbure ϵ_1 et ϵ_2 sont initialisés à 1. La superquadrique initiale est une simple ellipsoïde.

La position de la superquadrique (les paramètres de translation t_x, t_y, t_z) est initialement le centre de gravité du nuage de point.

$$\begin{aligned} t_x &= \frac{1}{N}\sum_{i=0}^{N} p_{x_i} \\ t_y &= \frac{1}{N}\sum_{i=0}^{N} p_{y_i} \\ t_z &= \frac{1}{N}\sum_{i=0}^{N} p_{z_i} \end{aligned}$$

L'orientation de la superquadrique est calculée avec la matrice M des moments centraux d'ordre 2:

$$M = \frac{1}{N}\sum_{i=1}^{N} \begin{pmatrix} (y_i-\bar{y})^2+(z_i-\bar{z})^2 & -(y_i-\bar{y})(x_i-\bar{x}) & -(z_i-\bar{z})(x_i-\bar{x}) \\ -(x_i-\bar{x})(y_i-\bar{y}) & (x_i-\bar{x})^2+(z_i-\bar{z})^2 & -(z_i-\bar{z})(y_i-\bar{y}) \\ -(x_i-\bar{x})(z_i-\bar{z}) & -(y_i-\bar{y})(z_i-\bar{z}) & (x_i-\bar{x})^2+(y_i-\bar{y})^2 \end{pmatrix} \tag{2.9}$$

En effet, la matrice de rotation R est simplement la matrice diagonalisant celle des moments centraux, c'est à dire la matrice des vecteurs propres de M.

$$M = R^{-1} * D * R \tag{2.10}$$

On peut aussi remarquer que le vecteur propre correspondant à la plus petite valeur propre de M n'est autre que l'axe d'inertie minimum (i.e. l'axe principal de la superquadrique).

La taille de la superquadrique (a_1, a_2, a_3) est préalablement la taille de la plus petite boîte englobant tous les points du nuage et orientée comme calculé précédemment.

$$\begin{aligned} a_1 &= \frac{1}{2}\left|\max_{i=1..N}(p_{x_i}) - \min_{i=1..N}(p_{x_i})\right| \\ a_2 &= \frac{1}{2}\left|\max_{i=1..N}(p_{y_i}) - \min_{i=1..N}(p_{y_i})\right| \\ a_3 &= \frac{1}{2}\left|\max_{i=1..N}(p_{z_i}) - \min_{i=1..N}(p_{z_i})\right| \end{aligned}$$

2.2.4 Étude de Boult et Gross

Boult et Gross se sont intéressés à la méthode de Solina et plus particulièrement au calcul de la mesure de l'erreur. Ils ont étudié trois différentes mesures. La plus simple:

$$E_1 = \sum_{i=1}^{N}\left[\hat{f}(x_i, y_i, z_i; \epsilon_1, \epsilon_2, a_1, a_2, a_3, t_x, t_y, t_z, r_x, r_y, r_z)\right]^2 \tag{2.11}$$

est la même évaluation que celle de Bookstein. E_1 ne prend pas le volume en considération et donc n'est pas satisfaisante.

Avec le contrôle du volume:

$$E_2 = a_1 a_2 a_3 E_1 \tag{2.12}$$

E_2 donne une trop grande part au volume ce qui a tendance à donner des superquadriques trop petites.

La troisième considère la distance euclidienne de chaque point du nuage $p_i = (x_i, y_i, z_i)$ et le point d'intersection q_i entre la droite passant par p_i et le centre de la superquadrique et la surface:

$$E_3 = \sum_{i=1}^{N} \|p_i - q_i\| \tag{2.13}$$

Si E_3 n'est vraisemblablement pas le plus court chemin d'un point à la surface, des calculs plus précis ne sont pas exploitables.

E_1 et E_2 sont inutilisables, mais, ils concluent que E_3 était plus avantageuse que l'équation 2.8 de Solina. Ce calcul d'erreur ne semble pourtant que très peu (jamais?) utilisé.

2.3 Reconstruction d'objets à l'aide de superquadriques

L'approximation des données par une superquadrique n'est pas généralement suffisante pour représenter des objets réels. Ces derniers sont en effet très peu souvent aussi réguliers et symétriques. Deux approches sont alors envisageables.

2.3.1 Reconstruction d'objets à l'aide d'une seule superquadriques

Une solution afin d'obtenir une représentation plus précise des données, après les avoir approximées avec une superquadrique est de déformer cette dernière.

Les superquadriques sont tout à fait adaptées aux déformations. C'était d'ailleurs l'un des thèmes de l'article introduisant les superquadriques dans l'informatique graphique [1].

C'est cette approche que choisit E. Bardinet. Il désire modéliser des données (nuage de points de R^3) de type médicale et plus particulièrement le ventricule gauche.

Il approxime, tout d'abord, ses données grâce à la méthode de Solina détaillée plus haut. Ensuite, il applique à la superquadrique obtenue une déformation de type FFD[4]. Ces transformations ont été introduites par Sederberg et Parry. Elles peuvent être vues comme une grille 3D (i.e. une "cage à écureuils"). Dans sa forme initiale, la superquadrique n'est pas déformée, mais ses noeuds peuvent être déplacés et, par là, déformer la superquadrique.

4. Free Form Transformation

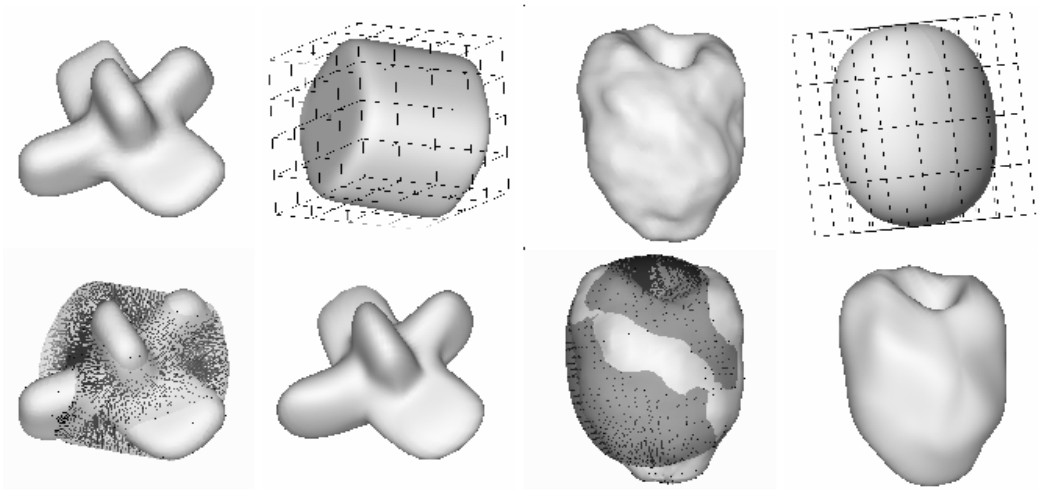

FIG. 2.4 – *Figure tirée de [2]. De la superellipsoïde au modèle final. À gauche donnée synthétiques, à droite données médicales. En haut: donnée et superellipsoïde après approximation. En Bas: Champ de déplacement entre les donnée et la superellipsoïde et le modèle final.*

Comme le montre la figure 2.3.1, cette approche donne de très bon résultats. Il faut néanmoins noter qu'elle s'applique surtout à des données assez proches de la superquadrique originale. En effet, dans le cas d'objets relativement complexes, la FFD risque au mieux d'avoir un nombre de coefficients très important (modèle non-compact) ou au pire ne pas fonctionner. Il faut aussi remarquer que cette approche ne favorise pas trop les comparaisons entre modèles, car l'objet n'est pas décomposé.

2.3.2 Reconstruction d'objets à l'aide de plusieurs superquadriques

Une autre approche est de considérer les données comme un assemblage de superquadriques. Tout le problème est alors la segmentation du nuage de point. Comment déterminer si un point appartient à un ensemble approximable précisément par une superquadrique?

De nombreuses méthodes, plus ou moins abouties, de segmentation de données tridimensionnelles existent, nous n'en aborderons une seule.

Leonardis-Solina : Region Growing

La méthode considérée par Leonardis et Solina [4, 5] est de type "region growing".

Des germes (petites régions de points) sont initialement placées suivant une grille dans le nuage de points. Chacun de ces germes est approximé par une

superquadrique. Cette approximation se fait, bien sûr, grâce à la méthode de Solina détaillée plus haut. Chaque germe essaie alors d'absorber, petit à petit, les points voisins (ces points voisins sont ceux qui lorsque qu'on fait grossir la superquadrique, "entrent" dans cette dernière). Si l'approximation de ce nouvel ensemble est convainquant, les points sont définitivement absorbés, sinon le développement est arrêté pour la région. On s'arrête lorsque toutes les régions sont stables. On obtient alors une partition des données et ainsi la décomposition en superquadriques de l'objet.

Un défaut important de la méthode est que partant de petites régions pour arriver à la décomposition (de locale à globale), on ne peut pas vraiment contrôler la précision. On est forcé de terminer si on ne veut pas se retrouver avec une multitude de superquadriques. A l'inverse, une décomposition global à local permet de totalement contrôler la résolution.

Chapitre 3

Méthode

3.1 Choix du modèle de surface

Comme on a pu le voir précédemment, les superellipsoïdes offrent une grande quantité de formes pour seulement cinq paramètres, mais ces dernières sont toujours très régulières et symétriques par rapport aux trois axes. Pour l'approximation d'objets réels, cela n'est pas vraiment un avantage, car la réalité est bien souvent moins équilibrée.

Ainsi, on peut remarquer dans les méthodes d'approximation utilisant une union de superellipsoïdes [5] que certain objets, paraissant pourtant très simples, nécessitent l'assemblage de plusieurs primitives. Dans un souci, de simplicité (en terme de nombre) du modèle cela est ennuyeux.

D'un autre coté, il n'est pas envisageable de réduire le nombre superquadrique si le l'approximation y perd toute sa qualité.

Une solution est d'ajouter une déformation au modèle sans que cela ne lui enlève sa spécificité de descripteur compact. Une autre contrainte est de garder au modèle une forme implicite et une forme paramétrique, afin de rester à l'aise, à la fois, pour l'approximation et pour l'affichage.

Une solution simple est que cette transformation ne se fasse que sur un seul des axes de l'ellipsoïde, on a par exemple:

$$\left(\left(\frac{x}{a_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y+f(x)}{a_2}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{y}{a_3}\right)^{\frac{2}{\epsilon_1}} = 1 \tag{3.1}$$

et

$$\begin{cases} X(\eta,\mu) &= x(\eta,\mu) \\ Y(\eta,\mu) &= y(\eta,\mu) + f(x(\eta,\mu)) \\ Z(\eta,\mu) &= z(\eta,\mu) \end{cases} \quad \begin{cases} \eta \in [-\pi/2, \pi/2], \\ \mu \in [-\pi, \pi] \end{cases}$$

En posant simplement $f(x) = p_1 x^2 + p_2 x$, on ajoute seulement deux paramètres au modèle utilisé habituellement, étendant de manière non négligeable les formes possibles.

Finalement, c'est ce descripteur surfacique que nous utiliserons. Il s'agit d'un modèle très généraliste et particulièrement compact (13 paramètres en

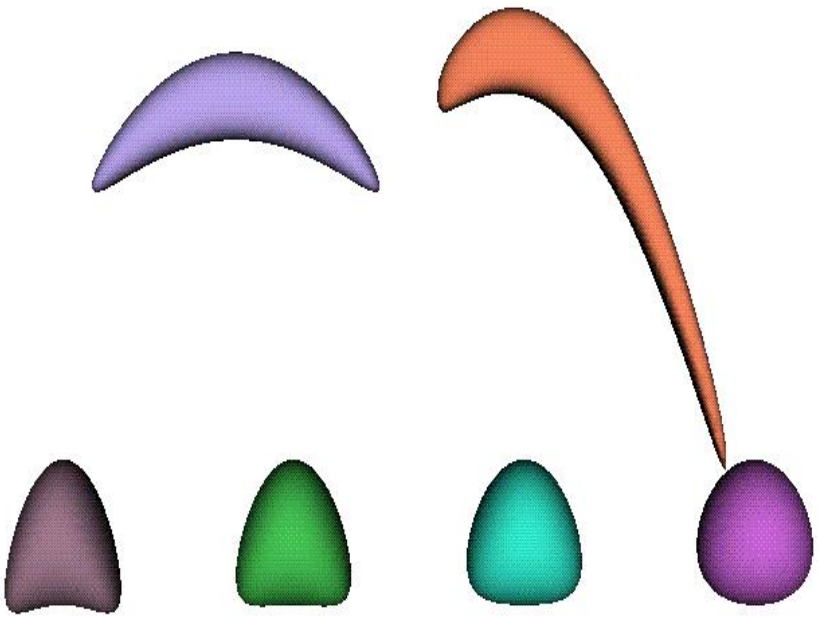

FIG. 3.1 – *Exemple de superellipsoïdes déformées. On peut remarquer que si au départ le but était d'obtenir les formes de la série (a), ce sont naturellement celles de la série (b) qui sont le plus souvent utilisé.*

comptant la déformation). On peut tout de même évoquer un léger défaut dû à cette modification : elle interdit l'usage des nombreux moteurs de rendu existant déjà pour les superquadriques.

3.2 Décomposition du nuage

Notre objectif est d'obtenir un assemblage de primitives modélisant de manière relativement précise un nuage de point de R^3.

Ayant fait le choix des primitives (les superellipsoïdes avec déformation), il faut maintenant une méthode de partitionnement des données tel que chaque sous-nuage s'approxime de manière idéale avec ce descripteur de base. Il va sans dire que l'approche ayant pour vocation un codage efficace (i.e. compact) cette méthode doit produire un nombre minimum de primitives pour une qualité d' approximation donnée.

3.2.1 Split and Merge

L'algorithme de décomposition utilisé est en fait un simple "split and merge". Cet algorithme est bien connu dans le cas de segmentation d'images 2D,

mais peut être utilisé dans de nombreux autres domaines.

La méthode se résume à deux étapes.

Première étape: Découpage du nuage (Split)

Le premier temps, de découpage, est récursif.

On essaie d'approximer le nuage entier par une seule superquadrique[1]. Si, suite à cette approximation, la somme du carré des distances de chaque point à la superquadrique relativisée au nombre de points:

$$\frac{1}{N}\sqrt{a_1 a_2 a_3}\sum\left(f\left(x_i, y_i, z_i; a_1, a_2, a_3, \epsilon_1, \epsilon_2, t_x, t_y, t_z, r_x, r_y, r_z, p_1, p_2\right)^{\frac{\epsilon_1}{2}} - 1\right)^2 \tag{3.2}$$

est inférieure à un seuil donné[2], alors le nuage reste intacte et la superquadrique résultante est considérée comme un modèle de ce dernier.

Dans le cas contraire, on divise le nuage.

Pour un split and merge classique (2D), les données sont alors généralement partagées en quatre, afin d'obtenir une structure de quadtree. En ce qui nous concerne, le nuage sera simplement scindé en deux. De plus, ce partage se fait de la manière la plus simple: on coupe par le plan orthogonal à un des axes passant par le centre de la région en traitement.

Suite à cette scission, on considérera par le même procédé les deux nuages résultants de facon totalement indépendante. Le mécanisme s'arrêtera de lui même, quand pour chaque région, le critère de split sera respecté.

On arrive donc, en fin d'étape, à un ensemble de régions parallépipédiques côte-à-côte contenants une partie distincte des données, dont une seule superquadrique suffit à approximer.

Il peut sembler que l'adaptation la plus fidèle de l'algorithme à la 3D eut été de découper le nuage en huit et de se ramener à une description en octree des données. Mais cela n'est pas une bonne idée compte tenu, en plus d'un nombre trop important de régions produites, des problèmes liés au découpage (3.2.2). De même, il aurait semblé peut être plus judicieux d'opérer un partage plus "intelligent", moins arbitraire; cette idée est évoquée plus loin (3.2.3).

Deuxième étape: Regroupement des régions similaires (Merge)

Il est vrai que la première étape fournit une partition des données, pour laquelle chaque région est associée à une superquadrique; mais il est impensable de s'arrêter là. Le découpage étant totalement arbitraire, le nombre de région (i.e. le nombre de superquadriques constituant le descripteur de l'objet) n'est pas du tout optimal et bien trop important.

Dans la seconde étape, on s'applique à regrouper certaines régions produites par la première étape.

1. Il s'agit bien sûr de superellipsoïde à laquelle on ajoute deux paramètres de déformation. Par souci de simplicité et de clarté, on préférera par la suite employer le terme plus générale de superquadrique ou plus simplement de superellipsoïde.

2. On peut parler de critère de split

Comme pour l'étape précédente, il faut déterminer un critère de fusion. Dans l'algorithme 2D classique, les deux critères sont généralement différents. Typiquement, on divise l'image tant que la variance des niveaux de gris (critère 1) n'est pas en dessous d'un certain seuil, et on fusionne deux régions si la moyenne de l'intensité des niveaux de gris (critère 2) est sensiblement équivalente.

Dans notre cas, on prend, pour les deux étapes, le même critère d'homogénéité. Les seuils, quant à eux, n'ont aucune raison d'être identiques[3].

Ainsi, on va appliquerà chaque couple (région, voisin[4] de cette même région), la procédure d'approximation à la fusion de leur nuages correspondants. Si cette approximation est satisfaisante (seuil) les régions seront regroupées, sinon, elles resteront indépendantes. Cette étape s'arrête lorsque la partition des données est stable.

Il faut, bien sûr, comprendre que ces tentatives de fusion ne se font pas uniquement sur des régions où le test avait déjà été fait lors de la première étape (grande minorité des cas qui de plus peuvent être facilement évités). En fait, pour chaque région, ce test n'a déja été fait qu'avec une seule voisine sur vingt six.

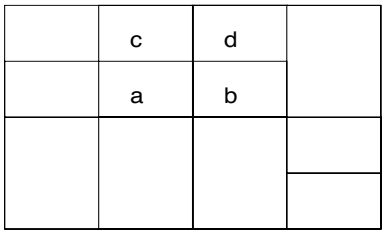

FIG. 3.2 – *Supposons que cette figure soit le résultat du découpage en régions de l'étape 1. Les région a et b n'ont jamais subi, entre elles strictement, le test d'homogénéité. En effet, ces deux parties des donné ont été séparées dès le premier découpage (i.e. en faisant le test sur le nuage entier). Le seul test réellement subi par a et b est , respectivement avec c et d.*

Notion de voisinage

La remarque précédente est encore plus vraie suivant la définition du voisinage que l'on a choisi. En effet, le voisinage direct (les régions ne sont séparées que par un plan, une arrête ou un sommet) est loin d'être la seule possibilité. Une autre solution peut être de considérer un voisinage basé sur la distance entre les régions (sont voisins les régions à distance inférieure à un certain seuil), ou bien un voisinage calculé suivant un degré de transitivité donné dans le graphe d'adjacence des régions. C'est cette deuxième voie que nous avons suivi.

3. Il est peut être même judicieux de prendre le seuil du critère de fusion supérieur au premier.

4. Ce terme reste encore à définir

En effet, considérer uniquement le voisinage direct limite trop les possibilités de fusion. Supposons que l'objet à décrire soit constitué globalement d'une superquadrique avec une "bosse". En prenant en compte un voisinage trop restreint, les régions autour de cette bosse ne pourront pas fusionner. Ce n'est pas le cas si la notion de voisinage est élargie.

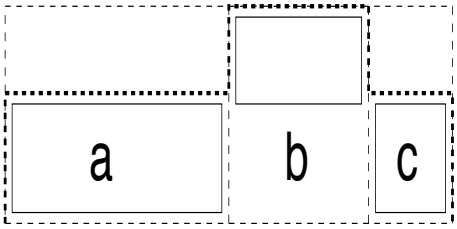

FIG. 3.3 – *En ne considérant que le voisinage direct les régions a et b ne peuvent pas fusionner puisque qu'elles ne sont pas voisines. En élargissant la notion de voisinage la barrière que représente la région b est levée.*

Il faut aussi noter que prendre un degré de transitivité trop élevé ralentit grandement l'algorithme : c'est un équilibre à trouver. On considère dans les applications un degré égal à deux (voisins directs et voisins de voisins directs).

A la fin de cette deuxième, et dernière, étape, l'ensemble des superquadriques, a été restreint au maximum pour la qualité d'approximation voulue.

3.2.2 Le problème "intérieur-extérieur"

Un gros problème de la décomposition, estcelui que l'on appellera "intérieur-extérieur" [5].

L'approximation d'un objet entier et donc d'une surface fermée ne pose en général pas ce problème. En fait, il survient lorsque l'approximation se fait sur une surface ouverte ce qui est le cas tout au cours de l'algorithme.

En effet, supposons qu'une région ne soit quasiment composée que de points appartenants à une partie concave, la superquadrique la plus proche sera assurément celle qui "remplit" cette concavité. Le problème est qu'il n'y a pas vraiment de moyen de savoir si une région ouverte est globalement une concavité ou une partie convexe.

En fait, ce problème est très dépendant de la facon de découper le nuage. Ayant choisi un découpage arbitraire, son apparition est très peu prévisible. S'il peut paraître courant en théorie, la pratique à montré qu'il était en fait assez rare.

Toutes les débuts de solution envisagées partaient de la connaissance au moins au départ de ce qui se trouvait à l'intérieur ou à l'extérieur de la surface à décrire. C'est vrai qu'il semblerait alors possible de reporter cette information lors du découpage. Il parrait même, qu'obtenir, au départ, l'information "intérieur-extérieur" pour un seul point serait suffisant. Mais comment obtenir

5. rien à voir avec la fonction du même nom 3.2.2

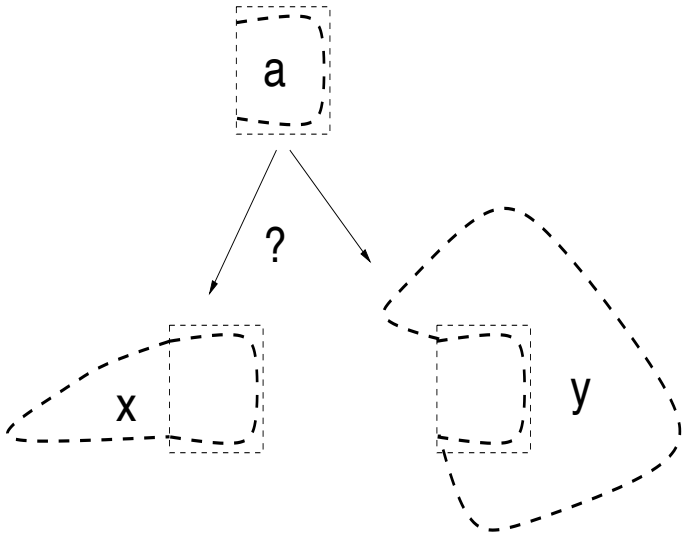

FIG. 3.4 – *Si au cours de la segmentation, on doit approximer la région a, la superquadrique résultante va "remplir" le nuage. C'est juste s'il s'agit d'une reg ion provenant de l'objet x mais totalement incorrect s'il s'agit de y*

cette information? Un maillage grossier pourrait peut-être aider. Mais, n'est ce pas un peu lourd?

Aucun article ne mentionne de ce problème. Néanmoins, la méthode de Leonardis ([5], [4]) devrait pourtant conduire à des choses similaires.

3.2.3 La question d'un découpage intelligent

La question d'un éventuel découpage plus "intelligent" doit tout de même être évoquée.

L'idée serait de détecter, d'une manière ou d'une autre, les composantes principales de l'objet, et de segmenter en conséquence. Cette décomposition, si habituelle pour un cerveau humain, n'est pourtant pas facile à réaliser.

Peut être, faudrait il rechercher les zones d'étranglement de l'objet ou les concavités. Cela nécessite une analyse particulièrement complexe et lourde du nuage.

Un avantage pourtant de ce genre de découpage serait de diminuer les cas d'apparition du problème intérieur-extérieur (3.2.2). D'un autre côté, l'orientation variable des régions compliquerait grandement l'analyse du voisinage, dont la conduite propre et efficace n'est déjà pas simple.

En définitif, ce genre de découpage est bien laborieux et peu robuste. De plus, l'approche considérée se veut particulièrement simple. On partitionne les données en ensembles de bases, puis on essaie de fusionner ces ensembles. Le découpage intelligent découle directement de l'algorithme sans analyse complexe des données.

Ne pas découper plus intelligemment ne vient donc pas du fait qu'on ne sait

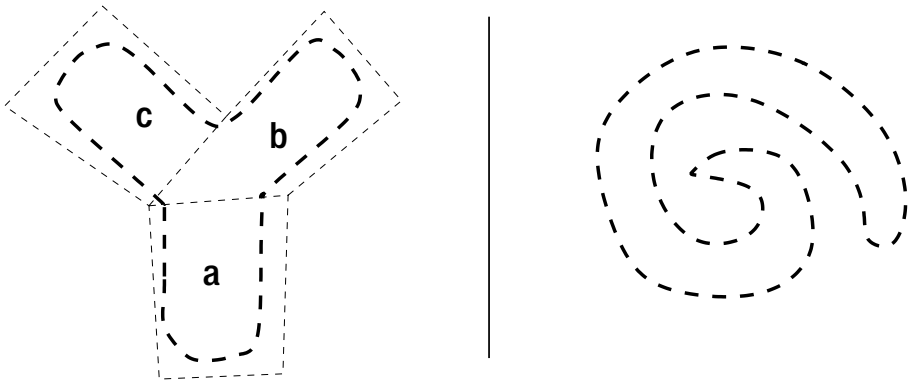

FIG. 3.5 – *A gauche: l'objet a été décomposé "intelligemment". Adroite : Quelle stratégie adopter?*

pas vraiment comment faire, mais parce que ce serait contraire à l'idée de la méthode.

3.3 Modèle final

Pour décrire un objet (en fait un nuage de points dans R^3), on pourrait en rester là. Le résultat serait un ensemble réduit non organisé de superquadriques. Mais, on peut aller plus loin et structurer la composition de cet ensemble.

La manière la plus évidente, et sans doute pas la pire, de faire cela est d'utiliser une méthode de type CSG[6].

Les superquadriques de l'ensemble résultant de la décomposition des données deviendraient les feuilles d'un arbres dont les noeuds seraient des opération ensemblistes (union, intersection et différence). L'état actuel des choses est en fait équivalent à un arbre ne contenant que des unions. Passer à l'étape supérieure (intersection et différence) nécessite de savoir résoudre le problème "intérieur-extérieur". En effet, les superquadriques posant problèmes dans 3.2.2 sont exactement celles qui doivent servir dans les différences[7]

Arriver à créer un tel arbre aurait au moins deux avantages.

Le plus évident est qu'il est bien plus simple de composer avec des différences pour bon nombre d'objets. En effet, pour tous les "objets à trous" par exemple, il est bien plus simple et surtout en terme de taille du modèle beaucoup plus avantageux d'opérer une soustraction que de "tourner autour du trou".

Le second avantage est qu'il serait possible, en analysant cet arbre de trouver et d'éliminer les redondances[8] dans la description de la scène (i.e. sous-arbres identiques par rotation, translation, facteur d'échelle...). La description de la

6. Constructive Solid Geometry

7. Mais bien sûr, tout le problème est qu'on ne sait pas quelles superquadriques posent problème.

8. Cela suppose bien sûr de savoir comparer deux modèle entre eux, mais cela est presque immédiat pour des arbres de superquadriques.

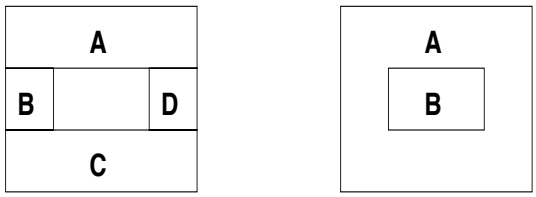

FIG. 3.6 – *Si on ne possède pas l'opérateur de différence, 4 primitives (dans le cas d'une décomposition parfaite!) sont nécessaire pour décrire un trou. Seulement 2 dans le cas contraire et la décomposition n'est pas très complexe.*

scène serait ainsi encore plus compacte. Notons qu'à ce moment là, l'arbre ne serait plus un arbre mais un graphe orienté acyclique (DAG). De plus, cela n'implique en rien de régler le problème "intérieur-extérieur".

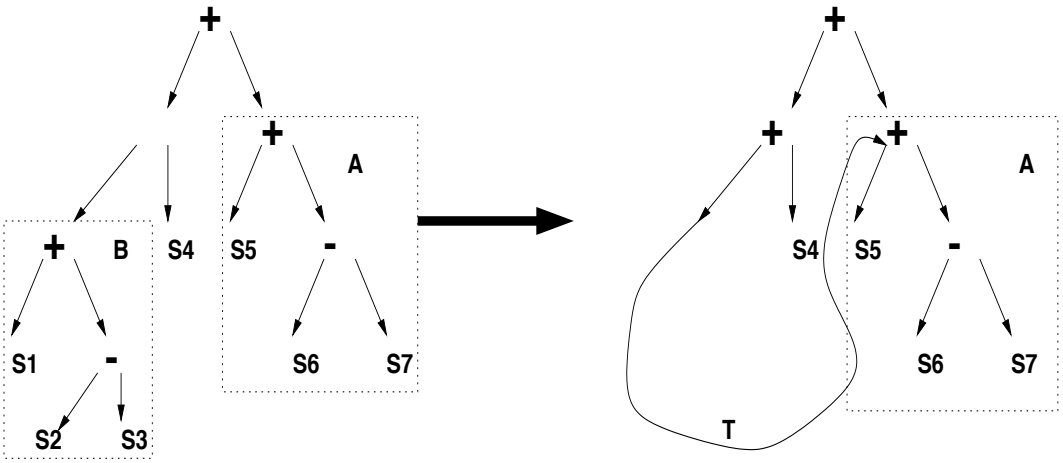

FIG. 3.7 – *Si la région A peut être considérée comme équivalente à T(B) alors le modèle peut encore être simplifié.*

Transmission du modèle

Les données 3D sont maintenant complètement modélisées. Il reste néanmoins, à brièvement évoquer le code à transmettre.

Il faudrait envoyer la structure du modèle (qui se résume en fait une suite d'opérations) et les superquadriques du modèle (13 paramètres).

Une méthode simple consistea en l' envoi de tout cela en préfixé (facile à analyser à la réception).

Exemple 1

$$M = Op_1 S_1 Op_2 Op_3 S_2 S_3 S_4$$

avec Op_i un opérateur et S_i une superquadrique.

Tout en sachant que pour Op_i trois valeurs seulement(on peut le coder sur 2 bits) sont possibles (union, intersection différence).

En ce qui concerne les superquadriques, on transmet les 13 paramètres, mais on peut diminuer la taille du code en jouant sur leurs précisions (notamment ϵ_1 et ϵ_2).

Ce mode de codage peut aussi être utilisé si on considère l'optimisation des équivalences entre sous-arbres évoqué plus haut. A ce moment là, il faut inclure la possibilité que S_i soit une superquadrique ou un couple identifiant-transformation.

Chapitre 4

Application

4.1 Reconstruction et visualisation du modèle

La méthode de modélisation décrite dans les parties précédentes occulte un aspect important du sujet. Obtenir la description compacte d'un objet est bien entendu l'étape de loin la plus complexe et la plus importante, mais la reconstruction et la visualisation du modèle ne doit pas être négligée. D'autant plus qu'une des contraintes de notre travail était que ces dernières soient très rapides.

Beaucoup de logiciels de modélisation ou de rendu (POV, Radiance, Lightwave...) supportent les superquadriques en standard et peuvent être utilisé pour la visualisation. Cependant leur utilisation nous est impossible suite à la modification de formule (déformation par le polynôme de degré 2). On doit donc utiliser une solution spécifique.

4.1.1 Virtual Reality Modeling Language (VRML)

Une première solution, compte tenu de l'aspect "transmission de données" du projet, est d'utiliser le langage qui à l'heure actuelle peut être considéré comme la norme de description de scène 3D sur Internet: VRML.

Cependant, si le VRML, permet grâce à peu de code d'afficher des primitives simple tel que des sphère ou des cubes ou certaines surfaces un peu plus complexes (B-splines), il ne sait pas gérer les surfaces définis par une forme paramétrique ou implicite générale.

Pour visualiser le modèle, à partir de l'arbre "ensembliste" de superquadriques, il est donc nécessaire de passer par une étape de facétisation de la scène. Bien entendu, il est hors de question de procéder à ce prétraitement avant la transmission réseau, et donc le poste "client" doit contenir une application spécifique pour le prétraitement. Le VRML perd donc tous ses avantages de norme de transmission de scènes.

4.1.2 Visualisation Toolkit (VTK)

Quitte à ajouter une application de prétraitement au poste client, autant que cette dernière s'occupe intégralement de la visualisation. Utilisé dans un

premier temps, VRML a été abandonné au profit d'une solution complètement originale : VTK.

VTK est une librairie de visualisation 2D/3D et de traitement d'images, elle est disponible gratuitement sur diverses plate-formes (UNIX, Linux, MS Windows...) et interfacable relativement simplement avec de nombreux langages (C++, TCL, Java, Perl...).

Son principal intérêt, en ce qui nous concerne, est de savoir traiter en standard les arbres ensemblistes de formes implicites générales. Cette librairie est donc particulièrement adaptée au problème. Pour preuve, il a suffi de surcharger la classe de surface implicite par notre classe de surface spécifique, pour pouvoir afficher (et manipuler : rotation, zoom ...) nos arbres d'union, intersection, différence de "superquadriques transformées" dans leur format original (i.e. sans prétraitement).

Appuyons le fait qu'en plus de gérer l'affichage et l'interactivité dans la scène, tous les calculs d'intersections, de différences, et d'unions sont gérés en standard par VTK pour toutes les classes dérivant de fonction implicite. Cela nous ôte une belle épine du pied car le calcul du rendu dans les zones de jonction des opérations booléennes entre superquadrique ne sont pas simples.

4.2 Approximation du nuage

Un algorithme d'approximation fiable et efficace est nécessaire pour le bon déroulement de la méthode. C'est en effet, à tous les niveaux qu'intervient le résultat de cette approximation. C'est pourquoi étudier différentes variantes n'est pas une perte de temps.

Dans tous les cas, c'est la méthode proposée par Solina qui sert de base. Les tests portent essentiellement sur l'algorithme de minimisation.

On passe par la première évaluation standard de paramètres, même dans le cas de minimisation globale (approche stochastique), où cela n'est théoriquement pas requis.

4.2.1 Approche stochastique : Algorithme Génétiques

Dans un premier temps, l'algorithme de minimisation standard (Levenberg-Marquardt) est remplacé par un algorithme génétique (méthode stochastique).

Fonctionnement d'un algorithme génétique

Les algorithmes génétiques sont fondés sur les lois darwiniennes de la sélection naturelle et peuvent être utilisés pour résoudre tout problème d'optimisation (minimisation, maximisation.....).

L'idée est de faire évoluer une population d'individus qui de génération en génération, après des reproductions, mutations, croisements, va sans cesse progresser[1] et ainsi s'approcher de la solution du problème.

1. C'est la sélection naturelle. Les individus faibles sont éliminés et les forts se reproduisent améliorant la population

Concrètement, il faut définir ce qu'est un individu. Dans notre cas, il s'agit simplement d'une superquadrique. Cet individu a des chromosomes, ici ce sont les paramètres de la superquadrique($a_1, a_2, a_3, \epsilon_1$...). Notre individu a donc 13 chromosomes. Il suffit alors d'avoir une fonction d'évaluation pour classer les individus de la population. Pour notre application, les superquadriques doivent être classées suivant leur valeur d'approximation du nuage, c'est à dire suivant 2.8. Cette fonction nous servira donc naturellement d'évaluation.

La première étape est alors, de créer aléatoirement une population de départ. Chaque individu se voit attribuer des paramètres au hasard.

Ensuite, débute le mécanisme itératif imitant les lois de sélection naturelle. Pour une génération donnée, on élimine les individus les plus faibles suivant la fonction d'évaluation. Les autres vont alors se reproduire pour donner la génération suivante. C'est durant la reproduction que se dessine l'amélioration. En effet, la reproduction se fait entre les individus fort (i.e. les superquadrique ayant les paramètres les plus proches des données). Durant celles-ci va s'opérer aléatoirement des croisements entre les chromosomes (moyenne sur les paramètres ou simple échange) mais aussi des mutations[2], où la valeur d'un paramètre est changé aléatoirement.

L'algorithme s'arrête soit au bout d'un certain nombre d'itérations (génération), soit quand la population ne s'améliore plus (quoiqu'on ne puisse pas vraiment savoir puisque des mutations, et par là une relance de progrès, peut intervenir à tout moment (car aléatoire)).

Résultats obtenus

Les algorithmes génétiques nécessitent un équilibre de paramètres assez subtil: taux de croisements, taux de mutations, nombres d'individus dans la population, nombre de générations... Mais quand celui là est atteint il donnent de très bon résultats.

Nous avons utilisé cette approche surtout pour l'étude de la fonction à minimiser (prise en compte du volume, déformation....) car elle était très souple. En effet, nulle besoin de connaître les dérivées de la fonction (celle-ci peut donc être modifiée très simplement), et pour ajouter un paramètre à ajuster, il suffit d'augmenter le nombrede chromozomes. De plus, ces paramètres peuvent être contraints très facilement (cela n'est pas normalement pas nécessaire (minimum global) mais l'accélération apportée est fortement appréciable).

Néanmoins, l'algorithme utilisé était assez lourd[3] et, après un bon rodage de la fonction d'évaluation, c'est un algorithme itératif classique et plus conventionnel qui aura notre préférence.

2. La mutation permet généralement de sortir des cuvettes, c'est à dire des minimums locaux

3. Ce n'est pas le cas de tous algorithmes génétiques qui peuvent être très efficace. Mais celui ci est rester assez généraliste, aucune optimisation pour l'application ou la fonction d'évaluation n'a été aparté

4.2.2 Approche itérative classique : Levenberg-Marquardt

Le second algorithme de minimisation utilisé est Levenberg-Marquardt, proposé par Solina et habituellement utilisé. C'est un algorithme qui converge rapidement et qui pour cela évolue entre une méthode quadratique et descente de gradient.

Le seul problème de la méthode est qu'il ne permet pas d'introduire de contraintes sur les paramètres. Hors, certains d'entre eux, ϵ_1 et ϵ_2 notamment, ne doivent jamais sortir d'un certain intervalle sous peine de résultats abérents. Certaines personnalisations ont donc dû être employées.

Contrainte des paramètres

Pour ajouter des contraintes nécessaires sur les paramètres, au lieu de changer un algorithme déjà bien rodé, c'est la fonction à minimiser (2.8) qui va être modifiée.

Les paramètres d'où proviennent le problème le plus fréquent liés à leurs non-contrainte sont a_1, a_2, a_3. En effet, théoriquement, la fonction

$$\sqrt{a_1 a_2 a_3} f^{\frac{\epsilon_1}{2}}$$

est sensée tendre vers l'infini lorsque a_1, a_2, a_3 tendent vers 0 et donc ne pas créer de minimum où l'algorithme pourrait converger. Pourtant, cela ne semble pas être le cas, ou le calcul numérique de la fonction pose certains problème qui n'apparaisse pas lors l'étude analytique des limites... Beaucoup d'exemples d'applications ont eu pour résultat une superquadrique ridiculement petite (parfois presque invisible après rendu) par rapport à aux données. Minorer ces paramètres était donc nécessaire.

On pose tout simplement:

$$a_i = amin_i + (a'_i)^2 \tag{4.1}$$

où $amin_i$ est constant et a'_i devient le paramètre à ajuster. Le minorant $amin_i$ a pour valeur 80 [4] de la valeur de a_i calculé lors de la première évaluation des paramètres. Les a_i sont donc contraints dans l'intervalle $[0.8 * aev_i, +\infty]$ avec aev_i la valeur de la pré-évaluation du paramètre a_i.

Le second souci vient des paramètres de courbure (ϵ_1 et ϵ_2). Pour eux, se pose un double problème. Ils sont forcément strictement positifs (sinon la formule, n'a plus rien d'une superquadrique) et doivent être majorés (2 ou 3 sont souvent utilisés comme majorants) car les grandes valeurs posent n'ont seulement des problèmes d'approximation et de calcul numérique, mais en plus donnent des formes complètement inintéressantes.

Il s'agit donc, contrairement à précédemment, de complètement encadrer ces paramètres. Pour cela, la solution choisie, est d'utiliser des fonctions bornées, typiquement cos ou sin. On posera donc:

$$\epsilon_i = \epsilon_{min} + (\frac{\epsilon_{max}}{2} - \epsilon_{min}) * (1 + \cos(\epsilon'_i)) \tag{4.2}$$

4. Cela n'est bien sûr pas forcément fixe.

avec ϵ_{min} (constante) la borne inférieure de l'intervalle et ϵ_{max} (constante) la borne supérieure. Le nouveau paramètre à ajuster est ϵ'_i.

Résultats obtenus

4.3 Résultats

Chapitre 5

Conclusion

Bibliographie

[1] Alan H. Barr. Superquadrics and angle preserving transformations. *a*, 1981.

[2] Eric Bardinet, Laurent D. Cohen, and Nicolas Ayache. A parametric deformable model to fit unstructured 3d data. *a*, 1995.

[3] Alok Gupta and Ruzena Bajcsy. Volumetric segmentation of range images of 3d objects using superquadric models. *a*, 1993.

[4] Ales Leonardis, Franc Solina, and Alenka Macerl. A direct recovery of superquadric models in range images using recover-and select paradigm. *a*, 1994.

[5] Ales Leonardis, Ales Jaklic, and Franc Solina. Superquadrics for segmenting and modeling range data. *a*, 1997.

[6] Alan H. Barr. Global and local deformations of solid primitives. *a*, 1984.

[7] Demetri Terzopoulos and Dimitri Metaxas. Dynamic 3d models with local and global deformations: deformable superquadrics. *a*, 1991.

[8] Ruud Bolle and Baba C. Vemuri. On three-dimensional surface reconstruction methods. *a*, 1991.