Echantillonnage anisotropique et rendu par points différentiels pour les surfaces implicites

Florian Levet, Julien Hadim, Patrick Reuter, Christophe Schlick

Echantillonnage anisotropique et rendu par points différentiels pour les surfaces implicites

F. Levet, J. Hadim, P. Reuter, C. Schlick

IPARLA project (LaBRI - INRIA Futurs) / Université Bordeaux 1, France
levet,hadim,preuter,schlick@labri.fr

Résumé : *Dans cet article, nous proposons une solution adaptant aux surfaces implicites, le rendu par points différentiels (differential point rendering) de Kalaiah et Varshney [KV01] originellement développé pour les surfaces paramétriques et les maillages triangulaires. La principale difficulté pour cette adaptation est que les deux étapes du processus d'échantillonnage proposé dans [KV01] s'appuient fortement sur des relations de voisinage entre les échantillons, voisinage qui n'existe pas naturellement pour les surfaces implicites. Pour résoudre ce problème, nous proposons d'étudier les possibilités d'extensions de la technique d'échantillonnage par particules de Witkin et Heckbert [WH94] afin de prendre en compte les directions et les valeurs des courbures principales de la surface implicite. Ainsi, nous présenterons des résultats provenant d'une utilisation de particules ellipsoïdales ainsi que les problèmes inhérents à la nature même de l'ellipsoïde. Enfin nous proposerons diverses directions de recherche dans le but de résoudre ces problèmes.*

Mots-clés : Modélisation géométrique, Surfaces Implicites, Géométrie Différentielle, Rendu par Points

1 Introduction

Les surfaces implicites sont une façon élégante de modeliser des surfaces 3D sans avoir à s'occuper de contraintes topologie, ce qui les rend intéressantes pour beaucoup d'applications graphiques. De plus, en utilisant les techniques de lancer de rayons pour visualiser les surfaces correspondantes, il est possible de développer un pipeline complet de modélisation-animation-rendu quasiment sans aucune contrainte de topologie. Malheureusement, afin de fournir un rendu intéractif de qualité raisonnable pour les surfaces implicites, il n'y a en général pas d'autre choix que de les convertir en maillages polygonaux, qui réintroduisent nécessairement de lourdes contraintes de topologie.

En 2001, Kalaiah and Varshney [KV01, KV03] proposèrent une technique innovante permettant d'obtenir une grande qualité de rendu des surfaces paramétriques. L'idée à la base de leur technique, appelée *differential point rendering*, est de générer un échantillonnage discret de la surface paramétrique, où chaque échantillon contient tous les paramètres qui définissent la géométrie différentielle locale (position, plan tangent, direction de courbure maximale et minimale, etc). Tous les échantillons sont alors rendus individuellement sans aucune information de connectivité, à l'intérieur d'un schéma de rendu par points qui utilise un "splatter" spécifique rectangulaire exploitant toutes les informations différentielles locales. Ce splatter peut être visualisé très efficacement en utilisant les vertex shaders des GPU programmables.

La technique de rendu par points différentiels est particulièrement bien adaptée aux surfaces paramétriques pour lesquelles il est vraiment efficace de générer un échantillonnage discret de points et de calculer tous les paramètres locaux de géométrie différentielle pour ces points. Les auteurs ont également proposé une extension directe de leur technique pour les maillages triangulaires, où les informations de géométrie différentielle peuvent être estimées par l'utilisation de plusieurs approches existantes, telles que celle développée par Taubin [Tau95] ou celle de Meyer et al. [MDSB02].

Au vu des propriétés intéressantes offertes par la technique de rendu par points différentiels, il est naturel d'essayer de l'étendre afin de pouvoir s'en servir avec les surfaces implicites. Malheureusement, comme nous allons le voir, cette extension n'est pas aussi évidente qu'elle pouvait apparaître à première vue. La technique d'échantillonnage proposée par Kalaiah et Varshney est basiquement un processus à deux étapes. Durant la première étape, un nuage de points relativement dense échantillonnant la surface est calculé, soit par un échantillonnage direct de l'espace paramètre (dans le cas d'une surface paramétrique) soit en utilisant les sommets existants (dans le cas d'un maillage triangulaire). Puis, durant la seconde étape, un grand nombre d'échantillons sont enlevés par un processus de simplification qui prend en compte les informations locales de géométrie différentielle (i.e. qui garde plus d'échantillons dans les directions de fortes courbures et moins d'échantillons dans les directions de faibles

courbures). A l'arrivée, on obtient un *échantillonnage anisotropique piloté par les informations de courbure* dans lequel chaque échantillon règne sur un domaine rectangulaire ou elliptique l'entourant, orienté localement en fonction des directions de courbures minimales et maximales. Le problème est que cette deuxième étape est fortement basée sur les relations de voisinage entre les échantillons. Cette relation existe naturellement pour les maillages triangulaires ou pour un échantillonnage régulier d'une surface paramètrique, mais elle n'a pas d'existence naturelle pour une surface implicite. En conséquence, l'ensemble du processus en deux étapes proposé par Kalaiah et Varshney s'adapte mal aux surfaces implicites.

L'objectif de ce papier est de présenter la solution alternative que nous proposons pour les surfaces implicites, permettant d'obtenir un échantillonnage anisotropique piloté par la courbure de la surface qui offre des propriétés similaires que celui de Kalaiah et Vershney. Cette solution est basée sur un système de particules, dans l'esprit de celui initialement proposé par Witkin et Heckbert [WH94]. La suite de ce papier est organisé comme suit : la section 2 présente quelques travaux existants principalement en relation avec les techniques d'échantillonnage de surfaces implicites. La section 3 détaille notre nouvelle technique échantillonnage anisotropique pour les surfaces implicites. La section 4 présente plusieurs résultats expérimentaux se focalisant plus particulièrement sur la qualité visuelle et sur les temps de convergence. La section 5 conclut et présente les nouvelles directions que nous sommes en train d'étudier. Finalement, des détails mathématiques concernant la géométrie différentielle pour les surfaces implicites sont donnés en Annexe.

2 Etat de l'art

Depuis les travaux fondateurs faits en 1985 par Levoy et Whitted [LW85], beaucoup de techniques de rendu par points différentes ont été développées au cours des ans, du fait que le sujet est devenu très populaire ces dernières années dans la communauté scientifique. Mais, comme notre objectif est d'adapter spécifiquement aux surfaces implicites la technique de rendu par points différentiels proposée par Kalaiah et Varshney [KV01], nous considérons qu'il serait hors de propos de rappeler dans cet article toutes les techniques existantes développées dans ce domaine. Un tour d'horizon très complet peut être trouvé dans le tutorial sur les techniques à base de points présenté à Eurographics 2002 [GPZ+02].

Dans cette section, nous allons nous pencher plus précisément sur les techniques proposées dans la littérature pour échantillonner les surfaces implicites. Les travaux existants peuvent être divisés en deux familles principales : les techniques de facettisation et les techniques à base de système de particules.

2.1 Techniques de facettisation

Les techniques de facettisation de surfaces implicites peuvent elles-mêmes être divisées en trois catégories. En premier lieu, les *techniques d'échantillonnage spatiales* subdivisent l'espace 3D en cellules, en général soit des cubes soit des tétraèdres, et recherchent les cellules qui intersectent la surface implicite. Une des plus connues est l'algorithme des "marching cubes" [WMW86, LC87], qui divise l'espace 3D en cellules cubiques, et des triangles sont générés en fonction du signe au coin des cellules. Les algorithmes des "marching tetrahedra", e.g. par Shirley et Tuchman [ST90] ou Hall et Warren [HW90], remplacent les cellules cubiques par des tétraèdres, ce qui permet d'éviter les configurations ambigües liées aux cubes. En contre-partie d'une meilleure robustesse les algorithmes des marching tetrahedra créent de nombreux triangles mal équilibrés.

La seconde catégorie est constituée par les *techniques de surface fitting* qui partent d'un maillage source qui approxime à peu près la surface implicite, maillage qui est ensuite adapté et déformé progressivement afin qu'il colle mieux à la surface implicite. Par exemple, Velho [Vel95, Vel96] commence avec une approximation polygonale brute de la surface en accord avec [GV92], et subdivise chaque polygone récursivement en fonction de la courbure locale. Toutes ces méthodes nécessitent une certaine connaissance de la topologie de la surface à partir du moment où l'approximation polygonale brute doit capturer la topologie correcte de la surface implicite. D'autres techniques de "surface fitting" ont été développées en se basant sur une recherche de points critiques [vOW93, BNvO96, SH97, WdGM99] mais elles souffrent d'une certaine inefficacité pour des surfaces implicites complexes.

Enfin les *techniques de surface tracking* commencent avec un élément de surface source et font grandir itérativement un maillage polygonal qui approxime la surface implicite. Les techniques de "cellular surface tracking"

[AG91, WMW86, Blo94] commencent à partir d'une cellule qui intersecte la surface implicite et, itérativement, trouvent toutes les cellules l'intersectant parmi ses voisines. Comme les cellules sont de taille constante, les techniques de cellular surface tracking souffrent des mêmes inconvénients que les techniques d'échantillonnage spatiales non adaptatives et, de plus, dans le cas général il peut être difficle de déterminer la cellule source.

2.2 Les techniques à base de particules

La seconde famille de méthodes d'échantillonnage de surfaces implicites est constituée par les systèmes de particules permettant de distribuer des échantillons de manière uniforme sur la surface implicite. Ce principe a été initialement introduit dans un outil complet de modélisation à base de particules orientées par Szeliski et Tonnesen [ST92], mais il n'y avait alors pas de surface implicite "sous" les particules. Figueiredo et al. furent les premiers à adapter un système de particules pour échantillonner des surfaces implicites [dFGTV92]. En appliquant un système de forces de répulsion sur les particules, les auteurs utilisèrent le processus de relaxation défini par Turk [Tur91] afin d'atteindre une distribution uniforme des points. Ces points sont ensuite utilisés pour calculer une approximation polygonale de la surface implicite.

Dans [WH94] Witkin et Heckbert ont assemblé tous ces travaux en une seule application de modélisation performante. En effet, ils montrèrent que les systèmes de particules sont très utiles afin d'afficher et de contrôler des surfaces implicites. Il utilisèrent deux sortes de particules : des *flotteurs* qui tiennent sur la surface afin d'avoir un bon rendu, et des *points de contrôle* pour déformer la surface. Ces deux types de particules doivent résoudre un ensemble de contraintes pour que les flotteurs suivent la surface implicite et que la surface suive les points de contrôle. De plus, les auteurs proposèrent un mécanisme de répulsion adaptative et des conditions de division/mort afin que les particules puissent soit se diviser, soit disparaître de la surface. Malheureusement aucune information sur la courbure de la surface implicite n'est prise en compte. Hart et al. [HBJF02] améliorèrent ce système de particules afin d'avoir une différentiation numérique et automatique de la surface implicite. En effet, dans les travaux de Witkin et Heckbert les calculs des dérivés utilisés par la répulsion peuvent devenir laborieux et générateurs d'erreurs. Encore une fois, la distribution des particules est uniforme, avec des particules sphériques qui ont toutes le même rayon de répulsion.

Dans [CA97] Crossno et Angel ont adapté les travaux de Witkin et Heckbert afin de pouvoir les appliquer dans les cas où l'on désire échantillonner une iso-surface extraite d'une image discrète 3D. Pour faire cela, ils utilisent une interpolation trilinéaire entre les huits sommets du voxel entourant le point échantillon pour obtenir une approximation de la fonction à cette position. Ils utilisent les mêmes forces de répulsion et calculs de mouvement des particules que dans [WH94] mais proposent d'estimer la courbure isotropique afin d'adapter le rayon de répulsion pour chaque particule en fonction de sa courbure. Les particules dans les régions de forte courbure ont un rayon de répulsion plus petit, et les directions de courbure principales ne sont pas prises en compte.

Finalement, Pauly et al. [PGK02] proposèrent un système de particules avec pour objectif de simplifier les surfaces à base de points. La même force linéaire que dans [Tur92] fut utilisée et appliquée sur une distribution de points calculée en fonction de la courbure de la surface MLS [Lev03]. L'utilisation d'une condition de mort combinée avec le schéma des forces de répulsion leur permirent de simplifier le nombre de points de la surface. Encore une fois, seules les informations de courbure isotropiques sont prises en compte.

3 Echantillonnage anisotropique

Comme dit dans l'introduction, notre objectif est de générer une technique d'échantillonnage anisotropique pour les surfaces implicites, qui offre des propriétés similaires à celle développée par Kalaiah et Varshney pour les surfaces paramétriques. Mais, en plus du fait que la densité de l'échantillonnage devra être reliée à la courbure locale, la technique devra également prendre en compte les directions de courbure minimale et maximale. En d'autres termes, chaque échantillon sera le centre d'un domaine ellipsoïdal orienté suivant ces directions et dont la taille dépendra des valeurs des rayons de courbure maximals et minimals. Pour atteindre ce but, nous proposons d'adapter la technique d'échantillonnage par particules développée par Witkin et Heckbert, rappelée dans la section précédente. Le coeur de l'algorithme d'échantillonnage (cf. algo 1) est similaire à celui de [WH94]. Dans cette section, nous allons détailler les choix faits pour chaque étape de cet algorithme.

Algorithm 1 Coeur de l'algorithme

Require: Une surface implicite
  Créer un ensemble initial de particules "collées" à la surface
  while La convergence n'est pas atteinte **do**
    Calculer les rayons de répulsion pour l'ensemble des particules
    Calculer les forces de répulsion pour l'ensemble des particules
    Mettre à jour les positions des particules
    Determiner quelles particules doivent se diviser
  end while

3.1 L'ensemble initial de particules

Grâce à l'étape de division des particules, l'algorithme finit par converger même lorsque l'on commence avec une seule particule initiale. Mais il est beaucoup plus efficace d'avoir un ensemble initial de particules qui couvre une large portion de la surface. La façon la plus simple d'arriver à ce résultat est d'utiliser un schéma similaire à la technique de *shrinkwrap* [vOW93] : échantillonner de façon régulière la sphère ou la boite englobante de la surface implicite, puis faire migrer les particules en bougeant les particules dans le sens contraire du gradient de la fonction implicite jusqu'à ce qu'elles atteignent la surface. Notez que le diamètre de la boite englobante est utilisé comme un facteur de normalisation d'échelle durant tout le processus, afin que toutes les distances (rayons, courbure, migration) soit calculées indépendamment de l'échelle.

3.2 Les rayons de répulsion

A chaque étape de la boucle de migration des particules, nous devons calculer les rayons de répulsion pour chaque particule. Comme dit précédemment, nous voulons un processus de répulsion anisotropique où les particules se repoussent plus dans les directions de faibles courbures et moins dans les directions de fortes courbures. Nous calculons donc actuellement deux rayons de répulsion par particules (un pour la direction de courbure maximale et un pour la direction de courbure minimale). Ceci est fait en adaptant pour les surfaces implicites, le calcul donné dans [KV01]. Les détails mathématiques de ce calcul peuvent être trouvés en Annexe. Il faut noter que ce calcul permet de déterminer la variation du champ implicite définissant la surface. Les valeurs trouvées ne sont donc pas des distances que l'on pourrait utiliser directement dans l'algorithme. Nous devons donc multiplier ces valeurs par un coefficient afin d'avoir des valeurs de distances utilisables comme domaine de répulsion autour de chaque particule. *minCurv* et *maxCurv* contiendront ces valeurs de courbure des deux directions principales *minCurvDir* et *maxCurvDir*.

3.3 Les forces de répulsion

Cette étape de l'algorithme diffère beaucoup du reste de la littérature. en effet dans [WH94, CA97, HBJF02] les auteurs calculent les forces de répulsion entre les particules en utilisant un calcul d'énergie. Si cela fonctionne très bien pour une répulsion isotropique, ce processus ne peut pas être généralisé pour des répulsion anisotropiques. Une autre façon de calculer les forces de répulsion entre particules a été proposée dans [Tur92, PGK02]. Là encore, le schéma de calcul est également proposé pour une répulsion isotropique entre des particules sphériques, mais contrairement aux autres, il est possible d'étendre ce schéma pour une répulsion anisotropique entre des particules ellipsoïdales.

Plus précisèment, une particule ellipsoïdale peut être définie en combinant les deux valeurs des rayons de répulsion *minRadius* et *maxRadius* et les deux directions orthogonales de courbure. Par l'ajout d'un troisième rayon dans la troisième direction (celle de la normale), une particule ellipsoïdale est définie. Nous utilisons des particules ellipsoïdale plutôt que des ellipses car, dans un espace 3D, il est plus simple de calculer des forces de répulsion entre des ellipsoïdes plutôt qu'entre des ellipses 2D qui n'appartiennent pas au même plan. Notez que le rayon donné pour la troisième direction (appelons-le la *hauteur* de la particule) ne compte pas vraiment : nous avons testé aussi bien avec des valeurs de hauteur très petites (afin d'avoir des ellipsoïdes presque plates) qu'avec des hauteurs égales à *minRadius* (afin d'avoir des particules avec une section circulaire) et les échantillonnages obtenu après migration des particules sont extrêmement similaires. La principale raison est que les barycentres des ellipsoïdes voisines sont presque dans un même plan durant les dernières étapes d'itération, et donc, les forces de répulsion sont plus ou moins orthogonales à la direction de la normale ce qui annule l'influence de la hauteur de la particule.

Une fois déterminées les dimensions de l'ellipsoïde, les forces de répulsion peuvent être calculées en fonction de l'algorithme donné ci-dessous. Il est important de noter que nous n'avons pas à calculer les forces de répulsion entre tous les couples de particules. En effet, une partition de l'espace est utilisée, qui permet d'éviter de calculer les forces entre deux particules qui appartiennent à des zones de la surface éloignées. Nous utilisons le schéma proposé par Witkin et Heckbert [WH94] mais n'importe quel partitionnement hiérarchique devrait bien fonctionner. En réduisant la complexité globale de $O(n^2)$ à $O(n \ln n)$, nous accélèrons de manière très importante les calculs impliqués dans le processus de migration.

Algorithm 2 Forces de répulsion entre deux particules

Require: Deux particules et leurs informations de courbure
Trouver le vecteur $\mathbf{r_{ij}} = \mathbf{p_j} - \mathbf{p_i}$ entre les deux centres des particules i et j
Trouver les deux angles α_i et β_i entre les deux directions de courbure de la particule i et $\mathbf{r_{ij}}$
Trouver les deux angles α_j et β_j entre les deux directions de courbure de la particule j et $\mathbf{r_{ij}}$
Trouver l'intersection $\mathbf{m_i}$ de $\mathbf{r_{ij}}$ et de l'ellipsoïde de i utilisant α_i et β_i
Trouver l'intersection $\mathbf{m_j}$ de $\mathbf{r_{ij}}$ et de l'ellipsoïde de j utilisant α_j et β_j
Determiner si les deux ellipsoïdes s'intersectent et calculer leur force de répulsion.

Nous notons $\mathbf{P_{imax}}$ (resp. $\mathbf{P_{imin}}$) le plan formé par les directions de courbure maximale (resp. minimale) de la particule i. Les angles α_i (resp. β_i) entre $\mathbf{r_{ij}}$ et $\mathbf{P_{imax}}$ (resp. $\mathbf{P_{imin}}$) sont définis par :

$$cos\alpha_i = \frac{\mathbf{P_{imax}}.\mathbf{r_{ij}}}{\|\mathbf{P_{imax}}\|\|\mathbf{r_{ij}}\|} \qquad cos\beta_i = \frac{\mathbf{P_{imin}}.\mathbf{r_{ij}}}{\|\mathbf{P_{imin}}\|\|\mathbf{r_{ij}}\|}$$

Et la position de l'intersection $\mathbf{m_i}$ entre l'ellipsoïde de la particule i et le vecteur $\mathbf{r_{ij}}$ est

$$\mathbf{m_i} = (maxRadius * \cos\alpha_i * \cos\beta_i, \quad minRadius * \cos\alpha_i * \sin\beta_i, \quad minRadius * \sin\alpha_i)$$

A partir des deux points d'intersection $\mathbf{m_i}$ et $\mathbf{m_j}$ l'intersection des deux ellipsoïdes est caractérisée par :

$$res = \|\mathbf{m_i}\| + \|\mathbf{m_j}\| - \|\mathbf{r_{ij}}\| \tag{3.1}$$

Si res est inférieur à 0 alors les deux particules i et j ne s'intersectent pas et elles n'appliquent aucune force sur l'autre. Mais si le résultat est supérieur à 0 une force de répulsion est appliquée sur la particule i par la particule j

$$F_{ij}(i) = res * (\mathbf{p_i} - \mathbf{p_j}) \tag{3.2}$$

Nous utilisons la même force de répulsion linéaire que dans [Tur92, PGK02] car son rayon de répulsion est fini. Le total des forces exercées sur i est donné alors par

$$F(i) = \sum_{j \in N_p} F_{ij}(i) \tag{3.3}$$

où N_p est le voisinage de i, tel qu'il est défini par le partitionnement spatial.

3.4 Migration des particules

Après avoir déterminé les forces de répulsion pour toutes les particules, la prochaine étape de l'algorithme est de déplacer les particules en fonction de ces forces de répulsion. Comme la force de répulsion $F(i)$ d'une particule i est un vecteur il suffit d'ajouter ce vecteur à la position courante de la particule afin de trouver sa nouvelle position : $\mathbf{p_i} += kF(i)$. La constante k définit la rigidité de la réaction de la particule en fonction des forces qu'on lui applique. Dans notre implémentation nous utilisons une valeur constante mais dans l'idéal, k devrait être proportionnel à la distance moyenne d'une particule avec ses voisines. Notez que, après ce mouvement, il y a des chances que

la particule ne soit plus “collée” à la surface mais qu’elle soit plutôt un petit peu à l’intérieur ou à l’extérieur de l’isosurface. Nous appliquons donc une étape de la méthode de Newton-Raphson afin de rapprocher les particules de la surface : $\mathbf{p_i} -= f(\mathbf{p_i})\mathbf{n_i}$, où $f(\mathbf{p_i})$ est l’évaluation de la surface implicite f au niveau de la particule i et $\mathbf{n_i}$ est la normale à la nouvelle position de la particule i.

Et, lorsque nous avons la position finale de la particule, nous devons calculer une dernière fois la normale afin d’avoir la bonne orientation de la particule pour le rendu.

3.5 Division éventuelle des particules

La dernière étape de notre algorithme est de déterminer si la particule doit être divisée ou non. Nous avons constaté qu’un bon critère est de subdiviser la particule lorsque la somme des normes des forces qui s’y appliquent devient inférieur à un seuil prédéfini. En effet, étant donné que les forces de répulsion sont appliquées dans un rayon fini autour des particules, le fait d’avoir une particule avec peu de forces appliquées signifie qu’elle se trouve dans une zone sous-échantillonnée. Il est donc naturel de la diviser pour augmenter la densité locale de l’échantillonnage.

3.6 Visualisation

Après que les caractéristiques des ellipsoïdes pour toutes les particules i aient été crées, nous nous en servons pour créer des points différentiels qui seront visualisés comme des rectangles ombragés. A partir du moment où les cartes graphiques actuelles ne supportent pas en standard les primitives courbées de type point différentiel nous avons tiré parti de la possibilité de programmer directement de nouvelles primitives au niveau des GPU de ces cartes graphiques.

Nous utilisons une primitive rectangulaire afin de représenter une particule dépendant de la géométrie différentielle locale. De façon similaire à [KV01], le rectangle est défini dans le plan tangent de la particule où la normale et les deux directions de courbure perpendiculaires définissent un système de coordonnées locales.

L’ampleur du rectangle est calculée en fonction des valeurs de courbures maximales et minimales. En conséquence, plus la surface est courbée et plus les rectangles seront petits. Afin de rendre le rectangle comme étant un morceaux lisse de la surface, une distribution adéquates des normales est nécessaire. Contrairement à Kalaiah et Varshney, qui sélectionnent la normal map qui convient le mieux par rapport à un ensemble de 256 normal maps précalculées dépendant des courbures principales, nous interpolons les normales par rapport aux coins du rectangle du point différentiel en utilisant les *vertex shaders* et *fragment shaders*. En effet comme nous connaissons la surface implicite qui sert de base à notre représentation, nous assignons une normale à chacun des quatres sommets du rectangle. En d’autres termes, nous définissons un champs de normales recouvrant le rectangle qui approxime localement l’apparence de la surface implicite. Ainsi, pour chaque sommet des coins du rectangle, le vertex shader copie la normale au niveau de ce sommet dans une coordonnée de texture. Ensuite, le fragment shader interpole les normales représentées par les coordonnées de textures pour chaque fragment et les normalise en utilisant une *cube map texture*.

Les rectangles sont des fragment-shaded utilisant le pipeline des cartes graphiques avec les programmes permettant de manipuler les vertex et les fragments. Dans le programme de vertex, nous ne calculons pas le shading car nous voulons une illumination pour chaque pixel. Nous écrivons la normale, la lumière et le demi-vecteur dans les registres de texture afin de les interpoler et de définir une distribution de normales dans l’espace écran. Dans le programme de fragment, nous normalisons les normales interpolées, la lumière et les demi-vecteurs et nous ombrageons les fragments avec les composants spéculaires et diffus en accord avec le modèle de Phong. A l’arrivée, l’ensemble de ces rectangles fragment-shaded qui se recouvrent donne une impression visuelle de surface lisse.

4 Résultats expérimentaux

Toutes les images présentées dans cette partie ont été visualisées avec un Pentium IV à 3.0 GHz équipé de 1 GB de RAM et d’une GeForce Quadro FX de NVidia. Aucun effort afin d’optimiser le code de gestion des particules n’a été effectué (la seule optimisation réalisée a été l’emploi d’une subdivision spatiale permettant de ne pas calculer toutes les forces de répulsion entre toutes les pairs de particules). Le rendu par points différentiels a été implémenté

en utilisant les vertex et fragment shaders de Cg avec une carte graphique NVidia équipée du chipset NV30.

Le seul paramètre que l'utilisateur peut modifier est le facteur d'échelle des valeurs de courbure. Notez que ce facteur doit être choisi avec soin. En effet, le rendu par points différentiels permet d'avoir un rendu lisse d'une surface implicite en utilisant un nombre réduit de points. Mais, si les rayons de répulsion sont trop grand, alors chaque point différentiel sera trop grand pour réaliser une bonne approximation de la portion de surface qu'il recouvre. La figure 1 souligne ce problème. En effet, la première image (1.a) montre une ellipsoïde rendue avec de gros rayons de répulsion. On peut voir des artéfacts de rendu au niveau de la silhouette de la surface, et la zone ombragée subie de l'aliasing. Contrairement à cela, l'image (1.c) montre une ellipsoïde rendue en utilisant de petits rayons de répulsion. Ici, la silhouette de la surface est bien rendue, avec aucun problème d'aliasing, et la zone ombragée est lisse. Le même phénomène est visible à une échelle plus faible sur les images du Bunny (1.b et 1.d).

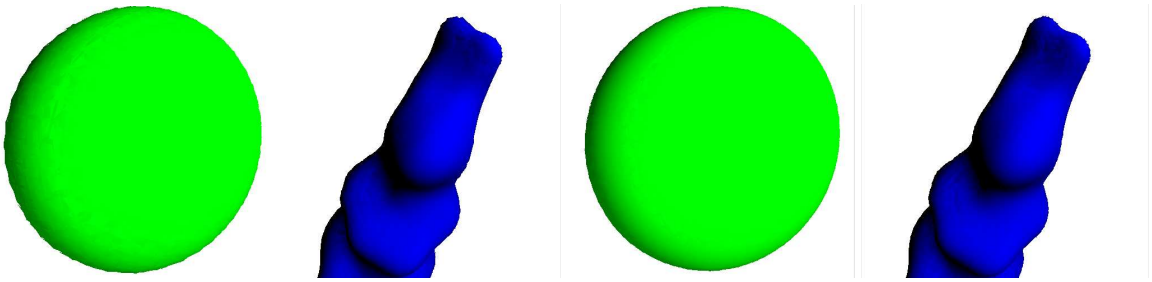

FIG. 1 – (a) Ellipsoïde à gros rayons (b) Bunny à gros rayons (c) Ellipsoïde à petits rayons (d) Bunny à petits rayons

Comme nous avons vu dans la section 3.6, le rendu par points différentiels est une collection de rectangles fragment-shaded qui se recouvrent eux-mêmes. Dans les figures 2.a et 2.b une couleur aléatoire a été utilisée pour chaque rectangle afin de mieux visualiser la structure sous-jacente. Enfin, à partir du moment où les dimensions et l'orientation du rectengle sont définies, n'importe quel fragment-shader peut être utlisé sans perte de performance. Les figures 2.c et 2.d montrent l'effet d'un rendu non photoréaliste obtenu par un simple changement de shader.

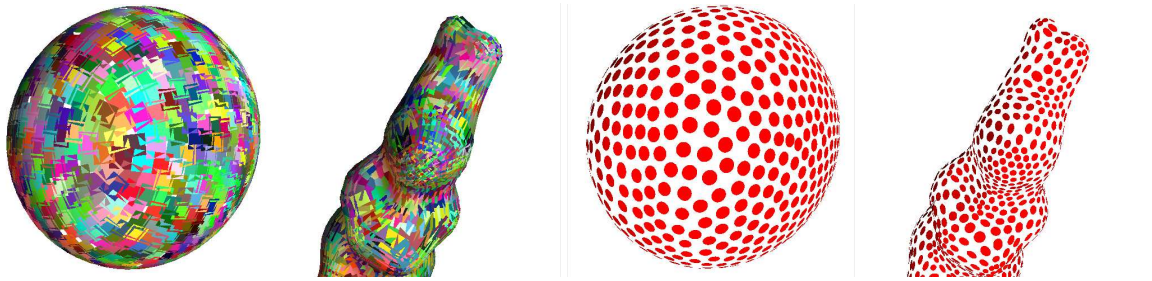

FIG. 2 – Rendu par points différentiels de couleurs choisies au hasard et rendu par particules de l'ellipsoïde et du Bunny

Le principal problème des systèmes par particules concerne la détection de la convergence; ainsi, même si la surface semble bien échantillonnée, on peut avoir une ou deux particules qui sont crées de temps en temps du fait du critère de division des particules. Il suffit qu'une particule se déplace très légérement en réponse à une force de répulsion pour faire basculer le critère de division et provoquer la création d'une nouvelle particule. Ces créations n'ont pas un gros impact sur le rendu du modèle mais elles montrent que la convergence des systèmes par particules est problématique. Un exemple de ce problème est donné par l'image 3.

Ces deux tableaux soulignent le fait que des particules sont crées même lorsque la surface est bien échantillonnée. Donc, le tableau 3.b montre qu'entre l'itération 600 et l'itération 800 il y a seulement 5 à 10 particules qui ont été crées. On peut donc dire que l'état d'équilibre du système a été atteint mais, à cause de la nature même des systèmes par particules, un nombre très faible de particules continue à se diviser. Un autre indice qui nous fait dire que l'état d'équilibre a été atteint est que, même si quelques particules ont été crées, le mouvement résiduel de toutes les particules montré à la fin du tableau a été très faible. Cela veut dire que la surface est bien échantillonnée et que les nouvelles particules ne sont pas très utiles, et qu'elles se déplaceront très peu et auront peu d'influence

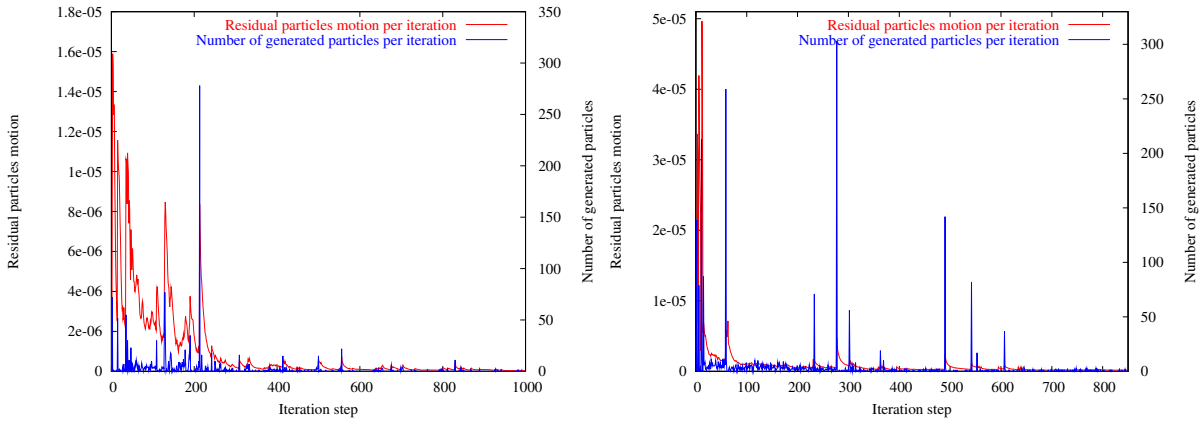

FIG. 3 – Mouvement résiduel des particules et nombre de particules crées par itération pour : (a) l'ellipsoïde (b) le Bunny

sur le mouvement des autres particules. Afin de résoudre ce problème, un critère plus robuste de convergence basé sur le mouvement résiduel des particules plutôt que sur un seuillage des forces devrait être mis en place.

5 Conclusion

Dans cet article, nous avons présenté une adaptation du rendu par points différentiels aux surfaces implicites en adaptant de manière originale le principe des systèmes par particules développés par Turk puis par Witkin et Heckbert. En liant ces deux approches, nous avons développé une façon élégante d'échantillonner et de visualiser des surfaces implicites.

Une des faiblesses de l'implémentation actuelle concerne la formalisation d'un critère robuste pour la détection de la convergence du système : dans le cas d'un système de particules sphériques qui ont toutes le même rayon de répulsion ou même des rayons de répulsion différents on peut compter sur un critère d'énergie pour identifier cette convergence. Mais, un critère d'énergie fonctionne bien pour tout ce qui n'est pas directionnel (comme les sphères), mais ne peut pas s'adapter à un système anisotropique comme les ellipsoïdes.

Nous pensons qu'une façon de réduire énormément le temps de convergence et le nombre d'itération du processus de relaxation serait de réaliser un premier échantillonnage avant de faire tourner le système par particules. En effet, ce premier échantillonnage serait fait afin d'avoir un certain nombre de particules qui serait suffisant pour échantillonner convenablement la surface, mais ces particules ne seraient alors pas à une place acceptable (beaucoup de particules qui s'intersecteraient) sur la surface. Alors, le processus de relaxation servirait à bien placer toutes les particules.

Et, enfin, nous pensons que les systèmes par particules sont parfaits pour aider à modéliser des surfaces implicites. Et donc, la prochaine étape de notre application sera de manipuler les particules directement sur la surface, et ensuite, la surface sera déformée afin de suivre les particules.

Références

[AG91] Eugene L. Allgower and Stefan Gnutzmann. Simplicial pivoting for mesh generation of implicitly defined surfaces. *Computer Aided Geometric Design*, 8(4) :305 – 325, 1991.

[Blo94] Jules Bloomenthal. An implicit surface polygonizer. *Graphics Gems IV*, pages 324–349, 1994.

[BNvO96] Andrea Bottino, Wim Nuij, and Kees van Overveld. How to shrinkwrap through a critical point : an algorithm for the adaptive triangulation of iso-surfaces with arbitrary topology. In *Proceedings of Implicit Surfaces '96*, pages 53–73, 1996.

[CA97] Patricia Crossno and Edward Angel. Isosurface extraction using particle systems. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization '97*, pages 495–498, 1997.

[Car76] Manfredo P. Do Carmo. *Differential Geometry of curves and surfaces*. Prentice-Hall, 1976.

[dFGTV92] Luiz Henrique de Figueiredo, Jonas Gomes, Demetri Terzopoulos, and Luiz Velho. Physically-based methods for polygonization of implicit surfaces. In *Proceedings of Graphics Interface '92*, pages 250–257. CIPS, 1992.

[GPZ+02] M. Gross, H. Pfister, M. Zwicker, M. Pauly, M. Stamminger, and M. Alexa. Point based computer graphics. In *Tutorial T6, Eurographics*, 2002.

[GV92] Jonas Gomes and Luiz Velho. *Implicit Objects in Computer Graphics*. Série Monografias do IMPA, Rio de Janeiro, 1992.

[HBJF02] John C. Hart, Ed Bachta, Wojciech Jarosz, and Terry Fleury. Using particles to sample and control more complex implicit surfaces. In *Proceedings of Shape Modeling International 2002*, pages 129–136, 2002.

[HW90] Mark Hall and Joe Warren. Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics & Applications*, 10(6) :33–42, 1990.

[KL96] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of ACM SIGGRAPH 96*, pages 313–324, 1996.

[KV01] Aravind Kalaiah and Amitabh Varshney. Differential point rendering. In K. Myskowski and S. Gortler, editors, *Rendering Techniques 2001 (Proceedings of the Eurographics Workshop on Rendering 2001)*, pages 139–150. Springer Verlag, 2001.

[KV03] Aravind Kalaiah and Amitabh Varshney. Modeling and rendering of points with local geometry. *IEEE Transactions on Visualization and Computer Graphics*, 9(1) :30–42, 2003.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes : A high resolution 3D surface construction algorithm. *Computer Graphics (ACM SIGGRAPH 87 Proceedings)*, 21(4) :163–169, 1987.

[Lev03] David Levin. Mesh-independent surface interpolation. In Guido Brunnett, Bernd Hamann, and H. Müller, editors, *Geometric Modeling for Scientific Visualization*. Springer, Heidelberg, Germany, 2003.

[LW85] Marc Levoy and Turner Whitted. The use of points as display primitive. Technical Report TR 85–022, University of North Carolina at Chapel Hill, 1985.

[MDSB02] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath '02*, pages 35–53, 2002.

[PGK02] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization 2002*, pages 163–170, 2002.

[SH97] Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proceedings of ACM SIGGRAPH 97*, pages 279–286, 1997.

[ST90] Peter Shirley and Allan Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics (San Diego Workshop on Volume Visualization)*, 24(5) :63–70, 1990.

[ST92] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics (Proceedings of ACM SIGGRAPH 92)*, 26(2) :185–194, 1992.

[Tau95] Gabriel Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Fifth International Conference on Computer Vision (ICCV'95)*, pages 902–907, 1995.

[Tur91] Greg Turk. Generating textures for arbitrary surfaces using reaction-diffusion. *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, 25(4) :289–298, 1991.

[Tur92] Greg Turk. Re-tiling polygonal surfaces. *Computer Graphics*, 26(2) :55–64, 1992.

[Vel95] Luiz Velho. Adaptive polygonization made simple. In *Proceedings of SIBGRAPI '95*, pages 111–118, 1995.

[Vel96] Luiz Velho. Simple and efficient polygonization of implicit surfaces. *Journal of Graphics Tools*, 1(2) :5–25, 1996.

[vOW93] C.W.A.M. van Overveld and B. Wyvill. Shrinkwrap : an adaptive algorithm for polygonizing an implicit surface. Technical Report 93/514/19, The University of Calgary, Calgary, Alberta, Canada, 1993.

[WdGM99] Shin Ting Wu and Marcelo de Gomensoro Malheiros. On improving the search for critical points of implicit functions. In *Proceedings of Implicit Surfaces '99*, pages 137–144, 1999.

[WH94] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of ACM SIGGRAPH 94*, pages 269–278, 1994.

[WMW86] Brian Wyvill, Craig McPheeters, and Geoff Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4) :227–234, 1986.

Appendice : Les Directions de Courbure Principales de la Surface Implicite

Dans la technique de rendu par points différentiels initiale [KV01, KV03], Kalaiah et Varshney proposèrent d'extraire les directions de courubres principales des surfaces paramétriques [Car76], des maillages triangulaires [Tau95], ou de surfaces NURBS qui sont ramenées à des maillages triangulaires [KL96]. Dans cette section , nous montrons comment extraire les directions de courbures principales pour un point $\mathbf{p} = [x,y,z]^T$ d'une surface implicite $\mathcal{S}$, i.e. $\mathbf{p} \in \mathcal{S}$. Pour cela, considérons la fonction $f : \Re^3 \rightarrow \Re$ définissant la surface implicite $\mathcal{S} = \{\mathbf{x} \in \Re^3 : f(\mathbf{x}) = 0\}$. Rappelons que la normale $\mathbf{n}$ d'un point $\mathbf{p}$ est défini par le gradient de cette fonction

$$\mathbf{n} = \nabla f(\mathbf{p}) = \left[\frac{\partial f}{\partial x}(\mathbf{p}), \frac{\partial f}{\partial y}(\mathbf{p}), \frac{\partial f}{\partial z}(\mathbf{p}) \right].$$

Afin de calculer la deuxième dérivée de la géométrie locale pour la détermination des directions de courubures principales , nous avons besoin de la *matrice hessienne* $\mathbf{H}$ des dérivées secondes de la fonction f :

$$\mathbf{H} = \begin{pmatrix} \frac{\partial \mathbf{n}}{\partial x}(\mathbf{p}) \\ \frac{\partial \mathbf{n}}{\partial y}(\mathbf{p}) \\ \frac{\partial \mathbf{n}}{\partial z}(\mathbf{p}) \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2}(\mathbf{p}) & \frac{\partial^2 f}{\partial x \partial y}(\mathbf{p}) & \frac{\partial^2 f}{\partial x \partial z}(\mathbf{p}) \\ \frac{\partial^2 f}{\partial x \partial y}(\mathbf{p}) & \frac{\partial^2 f}{\partial y^2}(\mathbf{p}) & \frac{\partial^2 f}{\partial y \partial z}(\mathbf{p}) \\ \frac{\partial^2 f}{\partial x \partial z}(\mathbf{p}) & \frac{\partial^2 f}{\partial y \partial z}(\mathbf{p}) & \frac{\partial^2 f}{\partial z^2}(\mathbf{p}) \end{pmatrix}$$

Nous utilisons une paramétrisation locale afin d'extraire les directions principales et les courbures au niveau d'un point $\mathbf{p} \in \mathcal{S}$ avec une normale associée $\mathbf{n}$ et une matrice hessienne $\mathbf{H}$. Pour illustrer les calculs qui suivent, veuillez vous référer à la figure 4.

Nous allons maintenant approximer la fonction f de la surface $\mathcal{S}$ autour d'un petit voisinage $\mathbf{p}$ en utilisant un petit vecteur $\mathbf{w}$ pour une expension de Taylor du second degré avec une erreur d'approximation de $o(\|\mathbf{w}\|^2)$:

$$f(\mathbf{p}+\mathbf{w}) = f(\mathbf{p}) + \mathbf{n}^T \bullet \mathbf{w} + \frac{1}{2}\mathbf{w} \bullet (\mathbf{H}\mathbf{w}) + (\|\mathbf{w}\|^2)$$

Comme $\mathbf{p} \in \mathcal{S}$, la fonction définissant la surface implicite $\mathcal{S}$ dans $\mathbf{p}$ est $f(\mathbf{p}) = 0$, et nous trouvons

$$f(\mathbf{p}+\mathbf{w}) = \mathbf{n}^T \bullet \mathbf{w} + \frac{1}{2}\mathbf{w} \bullet (\mathbf{H}\mathbf{w}) + o(\|\mathbf{w}\|^2).$$

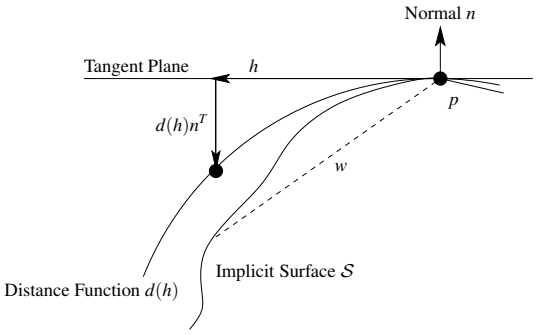

FIG. 4 – Calculer la courbure d'une surface implicite $\mathcal{S}$.

Maintenant, nous divisons le vecteur $\mathbf{w}$ en un vecteur $\mathbf{h}$ qui est orthogonal à $\mathbf{n}$, i.e. $\mathbf{n}^T \bullet \mathbf{h} = 0$, et en une fonction de distance d par rapport au plan tangent dans $\mathbf{p}$:

$$\mathbf{w} = \mathbf{h} + d(\mathbf{h})\mathbf{n}^T$$

Nous voulons déterminer la fonction de distance d afin que $f(\mathbf{p}+\mathbf{h}+d(\mathbf{h})\mathbf{n}^T) = 0$. En combinant les deux équations précédentes et en posant $d(\mathbf{h}) = o(\mathbf{h})$ car $d'(\mathbf{0}) = 0$ par définition, nous trouvons

$$\mathbf{n}^T \bullet (\mathbf{h} + d(\mathbf{h})\mathbf{n}^T) + \frac{1}{2}\Big((\mathbf{h} + d(\mathbf{h})\mathbf{n}^T) \bullet \big(\mathbf{H}(\mathbf{h} + d(\mathbf{h})\mathbf{n}^T)\big)\Big) = o(\|\mathbf{h}\|^2),$$

que nous développons en

$$\mathbf{n}^T \bullet (\mathbf{h} + d(\mathbf{h})\|\mathbf{n}\|^2 + \frac{1}{2}\Big(\mathbf{h} \bullet (\mathbf{H}\mathbf{h})\Big) + \frac{1}{2}\Big(\mathbf{h} \bullet \big(\mathbf{H}d(\mathbf{h})\mathbf{n}^T\big)\Big) + \frac{1}{2}\Big(d(\mathbf{h})\mathbf{n}^T \bullet (\mathbf{H}\mathbf{h})\Big) + \frac{1}{2}\Big(d(\mathbf{h})\mathbf{n}^T \bullet (\mathbf{H}d(\mathbf{h})\mathbf{n}^T)\Big) = o(\|\mathbf{h}\|^2).$$

Comme $\mathbf{n}^T \bullet \mathbf{h} = 0$, et comme nous pouvons négliger le terme constant par rapport à $\mathbf{h}$, nous trouvons

$$d(\mathbf{h})\|\mathbf{n}\|^2 + \frac{1}{2}\Big(\mathbf{h} \bullet (\mathbf{H}\mathbf{h})\Big) + d(\mathbf{h})\Big(\mathbf{h} \bullet (\mathbf{H}\mathbf{n}^T)\Big) = o(\|\mathbf{h}\|^2),$$

et l'approximation du second ordre de d est

$$d(\mathbf{h}) = -\frac{\mathbf{h} \bullet (\mathbf{H}\mathbf{h})}{2\|\mathbf{n}\|^2}.$$

Maintenant, nous voulons trouver le maximum et le minimum de $\mathbf{h} \bullet (\mathbf{H}\mathbf{h})$ pour $\mathbf{h}$ orthogonal à $\mathbf{n}$. Cela implique que la dérivée de $\mathbf{h} \bullet (\mathbf{H}\mathbf{h})$ aie une composante orthogonale à $\mathbf{n}$ avec la valeur 0, et donc que

$$\mathbf{H}\mathbf{h} = \mu\mathbf{h} + \lambda\mathbf{n}^T.$$

Pour déterminer les directions de courbures principales, nous devons trouver les eigenvecteurs, avec les eigenvaleurs différentes de 0 associées, de

$$\mathbf{H}\mathbf{h} - \frac{(\mathbf{H}\mathbf{h}) \bullet \mathbf{n}^T}{\|\mathbf{n}\|^2}\mathbf{n}^T = \big(\mathbf{I} - \mathbf{n}^T \bullet \mathbf{n}\big)\,\mathbf{H}\left(\mathbf{I} - \frac{\mathbf{n}^T \bullet \mathbf{n}}{\|\mathbf{n}\|^2}\right).$$

Et donc, les courbures principales $u_\mathbf{p}$ et $v_\mathbf{p}$ sont les eigenvaleurs différentes de 0 de cette matrice, et les directions principales $\mathbf{u_p}$ et $\mathbf{v_p}$ sont données par les eigenvecteurs correspondants.