Level of Detail Continuum for Huge Geometric Data

Florent Duguet, Carlos Henandez Esteban, George Drettakis, Francis Schmitt

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Level of Detail Continuum for Huge Geometric Data

Florent Duguet — Carlos Henandez Esteban — George Drettakis — Francis Schmitt

N° 5552

Avril 2005

Thème COG

ISSN 0249-6399

Level of Detail Continuum for Huge Geometric Data

Florent Duguet*† , Carlos Henandez Esteban † , George Drettakis * , Francis Schmitt †

Thème COG — Systèmes cognitifs
Projet REVES

Rapport de recherche n° 5552 — Avril 2005 — 25 pages

Abstract: In this paper we propose a unified solution for the creation of levels of detail on very large input data. We build a hierarchical signed distance function in an octree around the data and use this hierarchy to generate a continuum of levels of detail. Our distance function construction, based on the Gradient Vector Flow and the Poisson equation, builds on multigrid resolution algorithms. Using an appropriate interpolation scheme within the octree we obtain a continuous hierarchical distance function, which allows us to define a continuum of levels of detail for huge geometries. During this process, holes and undersampling issues in the input data are automatically corrected. We present three applications of our hierarchy: a novel hierarchical deformable model scheme that can automatically reconstruct closed Eulerian meshes of up to a million faces in a few minutes, an alternate distance-driven contouring approach, and raytracing of huge data models.

Key-words: Levels of detail, Poisson equation, distance fields, multigrid techniques, gradient vector flow, implicit surfaces, rendering

* REVES/INRIA Sophia-Antipolis
† GET / Télécom Paris CNRS-UMR-5141

Continuum de Niveaux de Details pour les Masses de Données Géométriques

Résumé : Dans ce papier nous proposons une solution unifiée à la création de niveaux de détails pour de très gros modèles. Nous construisons une distance signée hiérarchique dans un octree autour des données, et nous utilisons cette hiérarchie pour générée un continuum de niveaux de détail. La construction de fonction de distance, basée sur le GVF (Gradient Vector Flow) et sur l'équation de Poisson, repose sur des algorithmes de résolution multigrille. Une fonction de distance hiérarchique continue est obtenue en utilisant un schéma d'interpolation dans l'octree, ce qui nous permet de définir un continuum de niveaux de détail pour les très grosses géométries. Lors de ce processus, les trous et problèmes de sous-échantillonage des données en entrée sont automatiquement corrigés. Nous présentons trois applications à notre hiérarchie: un nouveau schéma de modèle déformable reconstruisant automatiquement des maillages eulériens fermés jusqu'à un million de polygones en quelques minutes, une méthode de reconstruction guidée par la distance, et un tracé de rayons de très gros modèles.

Mots-clés : Niveaux de Détail, Equation de Poisson, Champs de Distance, Méthodes Multigrilles, Gradient Vector Flow, Surfaces Implicites, Rendu.

2.6 k (0.7 s) 11 k (1.3 s) 43 k (3.4 s) 175 k (19 s) 700 k (2:10) 2,500 k (27:40)

Figure 1: Levels of detail on the entire St Matthew model, generated using the hierarchical deformable mesh reconstruction approach, driven by the proposed hierarchical continuous distance. The initial computation for this 370,000,000 points input model required 1h40. Below each image: number of triangles of the generated mesh (reconstruction time). Shading is done using per vertex ambient occlusion.

n this paper we propose a unified solution for the creation of levels of detail on very large input data. We build a hierarchical signed distance function in an octree around the data and use this hierarchy to generate a continuum of levels of detail. Our distance function construction, based on the Gradient Vector Flow and the Poisson equation, builds on multigrid resolution algorithms. Using an appropriate interpolation scheme within the octree we obtain a continuous hierarchical distance function, which allows us to define a continuum of levels of detail for huge geometries. During this process, holes and undersampling issues in the input data are automatically corrected. We present three applications of our hierarchy: a novel hierarchical deformable model scheme that can automatically reconstruct closed Eulerian meshes of up to a million faces in a few minutes, an alternate distance-driven contouring approach, and ray-tracing of huge data models.

1 Introduction

Nowadays, it is common to use acquired models of real objects to populate virtual environments. These models often come from 3D scanners or vision-based reconstruction techniques. They are typically represented as large point sets, which most often include normal information. For these representations to be useful in computer graphics applications, all operations available on traditional models, such as meshing, rendering, creation of levels-of-detail, etc., must also be provided. In addition, such point sets often lack information in certain parts of space, and thus require a hole-filling preprocess to be useable. Given the above, and the constantly increasing size of these models, it is clearly desirable to have a multi-resolution

representation of these data sets. Ideally, this representation should be fast to build, flexible and easy-to-use.

In this paper, we are concerned with the construction of a scale-space, i.e. a continuum of levels of detail for huge 3D models given as a set of points and oriented normals. If models are provided as polygonal or parametric surfaces, we can sample them in points and normals. To construct such a scale-space using previous techniques, it is necessary to combine several different approaches, and to perform three distinct steps. First, an object reconstruction step [CL96], followed by a hole filling phase [DMGL02], and finally a mesh simplification process [Hop96, HDD+93, GH97]. This three-step approach is cumbersome and costly since the maximal resolution object needs to be built first, and holes at this scale need to be filled. Finally, the simplification step needs to process the full resolution data, which can be extremely large [LPC+00]. For this process, all previous approaches, to our knowledge, use a volumetric approach [CL96, DMGL02, VCBS03, ZOF01]. Because of the size of the input, and the complexity of the problem, a hierarchical approach is necessary. Indeed, doubling the precision of the reconstruction requires eight times the number of cells for a regular grid.

We propose a unified solution to the creation of a scale space on very large data. We build a signed distance function, which we extend to a large volume around the object, arranged in an octree. The *signed* nature of this function enables us to easily perform automatic hole-filling and address undersampling issues in the input data. This distance is computed using well known partial differential equations (PDE): the heat and Poisson equations. Using this approach, we build on the vast literature addressing the problem of resolution of these equations (e.g. [Pop03, LGF04]).

Once this signed distance function is built, we can extract meshes of isosurfaces (using e.g. [Blo94, LC87, JLSW02]), and thanks to the continuity of our function, we can extract offsets for the same cost. We also have a guidance vector field which can be used for a projection operator. Finally, our distance function with its guidance vector provides us with a very well adapted force field for deformable models (snakes), improving the performances of previous methods [XP98, HS04].

Related Work

In the Multi-level Partitions of Unity (MPU) approach [OBA+03], the authors build a signed implicit function from the point and normals data sets. This technique includes hole filling, and computes a level-set close to the input surface. However, the construction algorithm - to its current extent - can only provide one level of detail. For other levels of detail (i.e. other error bounds in the approximation) the entire construction needs to be redone. Since the parameter of their construction controls the quality of the representation, the size of the output is not directly controlled.

Another similar approach, used in [CBC+01, TO02], defines an implicit function based on Radial Basis Functions (RBF). The function is constructed by selecting nodes from the input data, and solving a linear system to generate the RBF. The complexity of these data structures is directly related to the number of nodes, which are selected from the input data points using a specific pruning process. The precision obtained depends on this pruning

process, and is more flexible than a volumetric approach. However, this technique cannot handle huge input models easily; and the scalability in terms of output resolution is unclear.

Another approach, presented in [SOS04], describes an algorithm to define an implicit surface from a polygonal soup that can fit the input data. The approach produces a continuum of surfaces with increasing detail, but each surface requires the same amout of computation. Since the entire input dataset is used for the surface definition, this can be prohibitive for huge models.

In [ZOF01], the authors guide a level set with a distance function. The function is unsigned, and the convergence is reached when the normal to the level set surface is orthogonal to the gradient of the distance field (this criterion is similar to [AK04]). However, the implementation reported is for regular grids, and a hierarchical level set seems difficult to implement in this context.

In [VCBS03], the authors present a hole filling technique based on the joint diffusion of a scalar field and a vector field. This approach is related to ours in the idea of joint fields. However, they use regular grids and joint PDEs, whereas we use an octree and PDEs which are separable and which have very well known resolution schemes.

Others have used distance fields for various applications, but they either use unsigned distances [LPZ03] which do not provide a clear inside/outside predicate, or require closed models as input [FPRJ00, Gib98, JG96, Bae01, SOM04, HLC$^+$, HHVW96, SFYC96].

Contributions and Outline

The main contributions of our approach are: (i) the construction of a hierarchical signed distance function on an octree from huge input data, resulting in the rapid construction of a continuum of levels of detail, (ii) a novel hierarchical approach to surface extraction using deformable models which strongly accelerates previous techniques, and finally (iii) fully automatic hole filling on huge input data.

In what follows, we describe our algorithm to construct the hierarchical continuous distance field. After describing the hierarchical structure construction in Section 2, we define the continuous distance in Section 3 with an appropriate interpolation scheme, and present the continuum of levels of detail in Section 4. We show three applications of our approach in Section 5: a novel hierarchical surface extraction method using deformable models, an alternate distance driven mesh reconstruction approach, and ray-tracing of huge point sets. We conclude with a discussion of the merits and shortcomings of our proposed method.

2 Hierarchical Structure Construction

We want to define a signed distance function everywhere in space. We thus need to diffuse the only local information we have: points and normals on the surface. Computing the Euclidian distance in space can be achieved by solving the Eikonal equation:

$$|\nabla d| = 1, \tag{1}$$

however, the result is a function with many singularities [XP98]. Also, to our knowledge, this equation has no resolution scheme for signed distances on point sets with holes. We thus

propose another approach where we first diffuse a vector field and then extract a potential. This diffusion is thus done in two steps: the diffusion of the normal, as a vector field $\mathbf{u}$, and the computation of a scalar field d whose gradient is the vector field $\mathbf{u}$. The resulting field d, our distance function, does not correspond to the Euclidian distance, but it is very similar. Indeed, the gradient of this distance has fewer singularities than the gradient of the Euclidian distance. For example, if we consider a dihedral, the gradient of the Euclidian distance is undefined on the medial axis, and not continuous, where ours is smooth.

The initial knowledge we have for our distance function construction algorithm is the input set of points with oriented outgoing normals. Note that the gradient of the Euclidian distance on the surface is the unit normal to the surface. We first determine the cubic bounding box of the point set and scale it up to the volume where we want the distance function to be defined. We name this volume *domain box*.

The algorithm is then decomposed in three steps. The first step is an initialization where we build the octree from the input data. In the second step, we diffuse the normals in the octree using a Gradient Vector Flow (GVF). This approach based on Laplacian smoothing is used for 3D-snakes in [XP98, HS04]. It provides us with a smooth vector field approximating the Euclidian distance gradient field. We used an approach similar to [HS04] which operates on an octree. We further optimize it for higher convergence speed. After normalization, the output of this algorithm is a unit vector field noted $\mathbf{u}$ throughout the paper.

The third step uses the vector field $\mathbf{u}$. We determine the scalar field d such that the gradient of d is as close as possible to the vector field $\mathbf{u}$. This can be written as an energy minimization problem. Its solution can be obtained solving the Poisson equation. We use the input data as boundary conditions, where the scalar field is zero.

2.1 Octree Construction

We build the octree on the domain box. The subdivision is bounded by a user-defined level k_{max}. For each input sample (position p and normal), we subdivide the octree such that, at position p, the octree depth is k_{max}. As a result, all points are inserted at leaf nodes of level k_{max}. For numerical stability reasons, we also create a restricted octree: for each cell of level k, all the cells connected to it are of level $k-1$, k or $k+1$. This subdivision criterion can be satisfied using the following recursive algorithm:

```
ForceSplit (position,k)
  get leaf at position
  if leaf level < k then
    split leaf
    for each child center c
      ForceSplit (c,k)
  else
    for each neighbor center c
      ForceSplit (c,k-1)
```

We insert each sample (position and normal) in a leaf cell using the **ForceSplit** function with the sample position and k_{max} as input parameters, and we get the leaf cell of level k_{max} at this position. We then store the normal of the sample in this cell, and compute the distance of the center of the cell to the plane defined by the position and normal of the sample. These cells are further referred to as *data leaf cells*. If several samples are inserted in the same leaf cell, we count them and average the distances and normals. Finally, we unit the resulting normals.

This is a streaming process, and results in a data structure with output-dependent size. The actual samples are discarded as they are processed, and their influence (position and normal) is accumulated as part of the octree data.

This initialization phase is the only part of the algorithm that depends on the size of the input. Once all samples are inserted, the octree structure is frozen and the input data is no more used.

2.2 Diffusion of normals with the GVF

To define the distance function in each octree cell, we first diffuse the normal to generate a smooth vector field. We use the Gradient Vector Flow (GVF)[XP98], an algorithm applied to minimize the following energy functional:

$$E = \int \mu |\nabla \mathbf{u}|^2 + |\nabla f|^2 |\mathbf{u} - \nabla f|^2, \tag{2}$$

where f is an input scalar field, $\mathbf{u}$ is the computed vector field, and μ the weight of the regularization term. We will use the input unit normals instead of ∇f, and minimize the

energy functional:

$$E = \int \mu |\nabla \mathbf{u}|^2 + |\mathbf{n}|^2 |\mathbf{u} - \mathbf{n}|^2, \tag{3}$$

with $\mathbf{n}$ a fixed vector field defined as the input normals everywhere they are available, and zero elsewhere.

Since we do not want to smooth the input data[1], we use the iterative scheme (values at the center of the octree leaves):

$$\mathbf{u}_{t+1} = \mathbf{u}_t + \mu(1 - |\mathbf{n}|)\Delta \mathbf{u}_t, \tag{4}$$

where Δ is the usual discrete Laplacian operator, and $\mathbf{u}_t$ is the vector which evolves in this process. We initialize it with $\mathbf{u}_0 = \mathbf{n}$. According to Eq. (4), $\mathbf{u}_t$ fits the input normals at the center of the data leaf cells. Since $|\mathbf{n}| = 1$, we have $\mathbf{u}_{t+1} = \mathbf{u}_t = \mathbf{u}_0$.

For this iterative scheme, we use a multigrid approach which is known to be very efficient with an octree structure, as shown in [Pop03]. Results are shown in Figure 2.

Once convergence is reached, we normalize the vector field $\mathbf{u}$ to get a unit normal field.

2.3 Discrete Signed Distance

Our aim is now to define a distance value for the center of each octree cell. Locally around the surface, this distance function d is defined by the distance to the tangent plane, and we have:

$$\nabla d = \mathbf{n} = \mathbf{u}. \tag{5}$$

In order to extend this equation to the complete volume, we minimize the following energy functional:

$$E = \int |\mathbf{u} - \nabla d|^2. \tag{6}$$

This minimum is known to be obtained by the solution of the Poisson equation:

$$\Delta d = \nabla \cdot \mathbf{u}, \tag{7}$$

where $\nabla \cdot \mathbf{u}$ is the divergence of the vector $\mathbf{u}$.

This can be solved in a variational framework by applying an iterative scheme similar to Eq. (4):

$$d_{t+1} = d_t + \frac{h}{6}\left[\Delta d_t - \nabla \cdot \mathbf{u}\right], \tag{8}$$

where h is the size of the cell and d_t is the distance function (with a value at the center of the cell) evolving during the resolution of the equations. Note that convergence is reached when Eq. (7) is satisfied. We use a multigrid resolution scheme as presented in [Pop03], along with the associated numeric scheme. Results are shown in Figure 3, and performances in Table 1.

[1]The basic iterative scheme for the gradient vector flow, as in e.g. [HS04] is given by: $\mathbf{u}_{t+1} = \mathbf{u}_t + \mu\Delta\mathbf{u}_t - (\mathbf{u}_t - \nabla f)|\nabla f|^2$, which actually smoothes the input data.

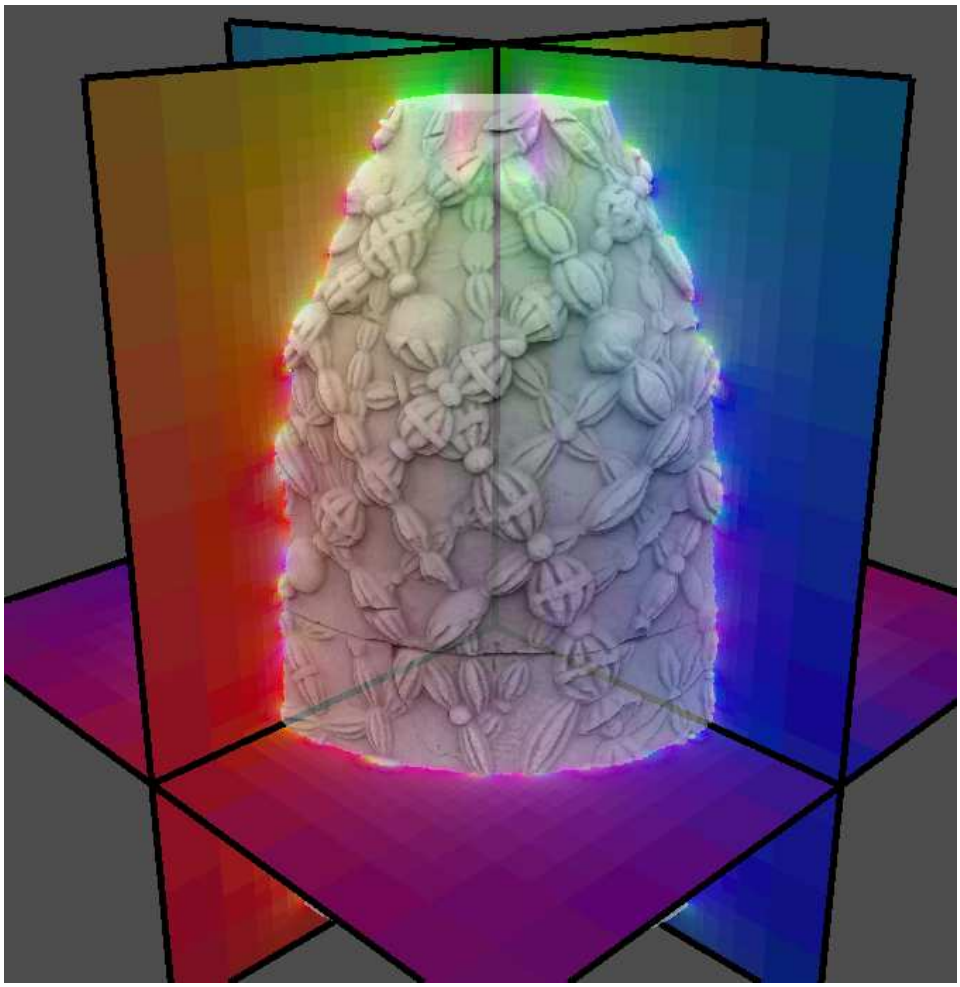

Figure 2: Diffusion of normals: result of the GVF algorithm for octree level 10 on the Omphalos model (input: 11,500,000 samples). The vector field is color-coded using usual false colors (r=x, g=y, b=z). Note that the model is rendered semi-transparently on the cuts of the field.

As in Section 2.2, the initalization values and boundary conditions are given by the data leaf cells. In the octree construction step, we computed the distance of the center of the data leaf cell, and this initial value for data leaf cells are used as boundary conditions and will not change. The other cells are initialized with $d = 0$.

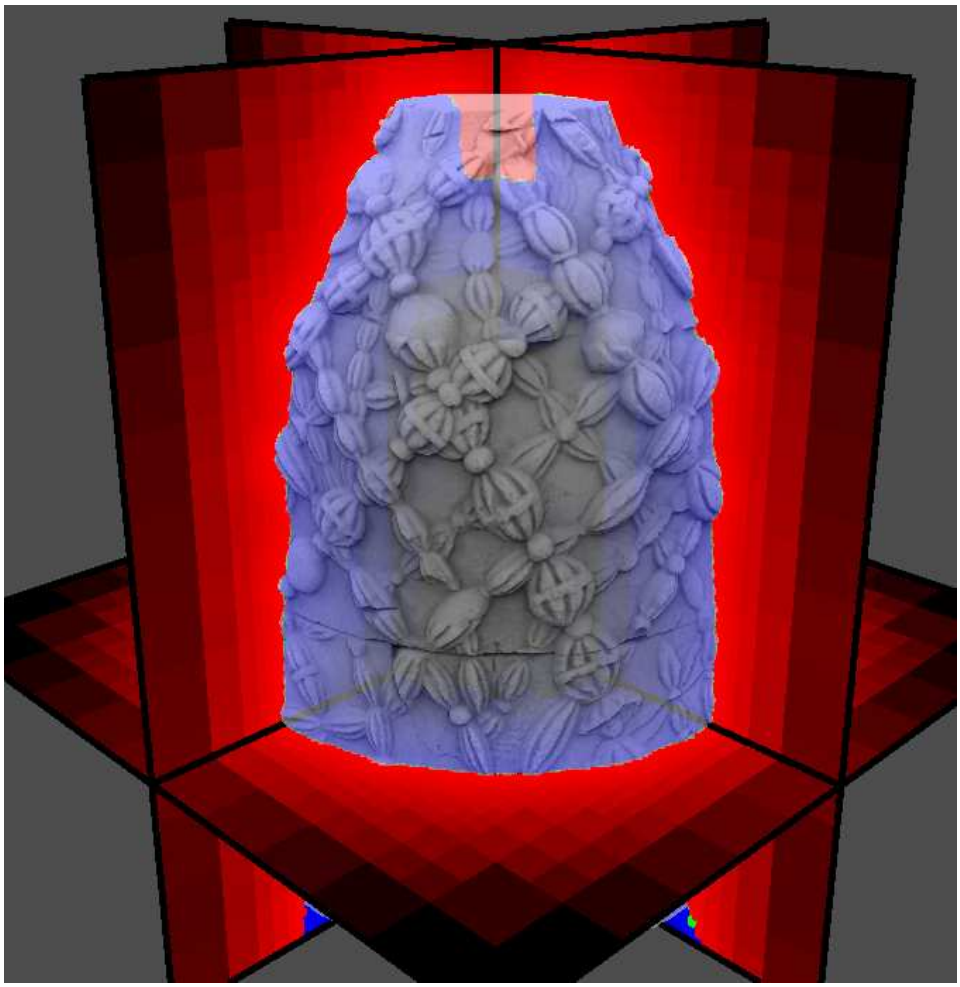

Figure 3: Discrete signed distance: result of the Poisson equation resolution algorithm (octree level 10, Omphalos model) applied on the diffused normal field shown in Figure 2. Red is outside (deeper is further), blue is inside and green is where the distance is zero.

3 Distance Function

The octree contains samples of the distance and its gradient only at the center of the cells. To obtain a continuous distance field everywhere we use two approximation schemes of these functions, first described on a regular grid, and then extended to an octree.

Object (size)	k_{max}	*init*	*GVF*	*Poisson*	*Tot.*
Bunny (70 k)	7	0.17	8.2	12.9	0:21
Armadillo (350 k)	8	1.07	25	39	2:11
Buddha (1.0 M)	9	2.67	1:37	1:33	7:17
Omphalos (11.5 M)	10	27	15:16	24:48	0h39
Dancers (47.9 M)	11	4:24	48:57	55:21	2h49
StMatthew (370 M)	11	45:46	29:30	24:52	1h40

Table 1: Performances for octree construction algorithm. Platform is a Pentium IV, 2.8 GHz with 1GB RAM, and an IDE hard drive. The columns are: object name (number of input points, k stands for thousands and M for millions); k_{max} is the octree depth; *init*, *GVF* and *Poisson* are the times (in hours(h)minutes(:)seconds) for the initialization step, the normal diffusion and the Poisson equation resolution respectively. *Tot.* is the total construction time for all steps. For very large models, *init* can be up to 2/3 for loading the data and 1/3 for processing it (e.g. St.Matthew: 30 min loading and 15 min processing).

For these schemes, we define a *prediction distance* for each cell. The prediction distance $\hat{d}^{(i)}(\mathbf{x})$ of cell i at any position $\mathbf{x}$ is defined by:

$$\hat{d}^{(i)}(\mathbf{x}) = (\mathbf{x} - \mathbf{c}^{(i)}) \cdot \mathbf{u}^{(i)} + d^{(i)}, \tag{9}$$

where $\mathbf{u}^{(i)}$ (resp. $d^{(i)}$) are the guidance vector (resp. distance value) at the center $\mathbf{c}^{(i)}$ of cell i. This equation can be seen as a first order Taylor approximation of the distance d around the center $\mathbf{c}^{(i)}$. This prediction distance can be evaluated either inside or outside the cell.

We can define a coarse interpolation scheme with the prediction distance. For a given position $\mathbf{x}$, we get the prediction distance of the leaf cell at $\mathbf{x}$. This interpolation scheme is fast to evaluate, but is not continuous.

3.1 Regular Grid

Let us consider first the case of a regular grid. For a given position in space, we consider a cubic box $\mathcal{B}_\alpha$ aligned with the axis, and of constant size s defined as follows:

$$s = (1 + \alpha)h,$$

where h is the size of the cell, and α is a positive scalar. We then compute a weighted average of the distance predicted (according to Eq.(9) by the cells hit by this box:

$$d(\mathbf{x}) = \sum_j w_j(\mathbf{x}) \hat{d}^{(j)} = \sum_j w_j(\mathbf{x}) \left[d^{(j)} + (\mathbf{x} - \mathbf{c}^{(j)}) \cdot \mathbf{u}^{(j)} \right], \tag{10}$$

where weights w_j are given by the volume of intersection between the cells j and the box $\mathcal{B}_\alpha$. Note that for $\alpha = 0$, the approximation scheme becomes an interpolation scheme. An illustration of the weights in 2D are shown in Figure 4 (top row).

3.2 Extension to Octree

We extend this approach to the octree by the definition of a size parameter, smoothly varying within the domain box. Let k be the level of the cell corresponding to the position of query $\mathbf{x}$. If all the neighboring cells are either of level k or $k-1$, the size parameter is given by the previous equation. Otherwise, getting closer to a cell of higher level, the size is smoothly reduced. Let δ be the distance of $\mathbf{x}$ to the closest neighbor cell of level $k+1$. We define $\theta = min(2\delta/h_k, 1)$. Then, by using a cubic Hermite interpolation, we define the following $\mathbf{C}^1$ size parameter:

$$s(\mathbf{x}) = (1+\alpha)\left[\frac{1}{2} + \theta^2(\frac{3}{2} - \theta)\right] h_k, \tag{11}$$

where h_k is the size of the cell at level k. An illustration of the weights in 2D is given in Figure 4 (bottom row).

In practice, we chose $\alpha = 0$ in order to preserve details best. Increasing the value of α will progressively smooth away details. Results of this interpolation scheme are shown in Figure 5. We can use the same approximation scheme for the guidance vector $\mathbf{u}$ when required.

4 Level of Detail Continuum

Now that we have computed a continuous distance field $d(\mathbf{x})$ everywhere in our domain box, we can build several levels of detail of this function. The distance function defined with this octree and the interpolation scheme is noted d_n, where $n = k_{max}$ is the working level (maximal depth of the octree). We define values at inner nodes of the octree as the average of the values of its children. We define the other levels for the distance function of the octree using node or leaf values up to a given level. The distance function at level k is defined by the octree using cells of level 0 up to k. If a cell at level k is not a leaf, we use its node value. We thus define a discrete set of distance functions $d_1..d_n$.

We can define a continuum of levels of detail by blending distance functions of two successive levels. For example, the distance at level 7.5 can be defined as the average of the distance at level 7 and the distance at level 8. More formally, let λ be the level value $\lambda \leq k_{max}$, we define d_λ as:

$$d_\lambda = (\lambda - \lfloor\lambda\rfloor)d_{\lfloor\lambda\rfloor+1} + (1 - (\lambda - \lfloor\lambda\rfloor))d_{\lfloor\lambda\rfloor}, \tag{12}$$

where $\lfloor\lambda\rfloor$ is the integer part of λ.

We have thus defined a continuum of surfaces along two axes: the offset axis, and the detail axis. Note that these two axes are different. Surfaces can be defined using these two parameters:

$$\mathcal{S}(\lambda, \mu) = \{\mathbf{x} : d_\lambda(\mathbf{x}) = \mu\}. \tag{13}$$

For a given level λ, we can choose different offsets. An example is shown in Figure 7.

The following section presents how such a continuum of levels of detail can be efficiently applied in practice.

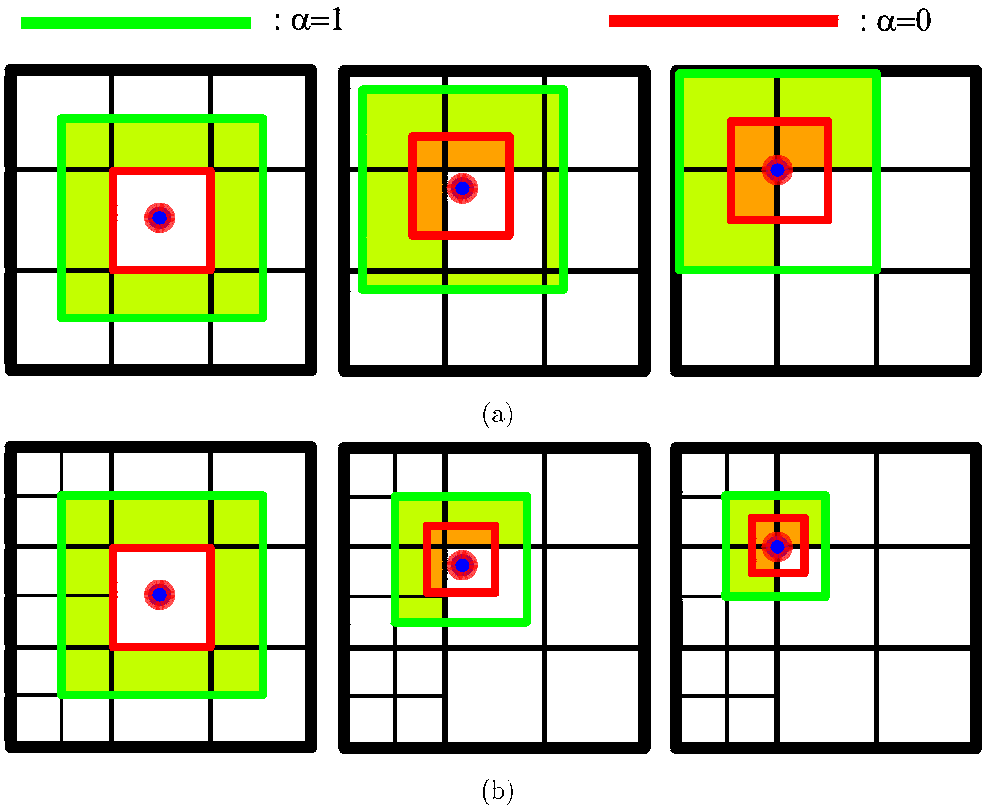

Figure 4: 2D illustration of the interpolation weights for two values of α. Top: for the grid; Bottom: for the octree. Left: for the center of the cell; Middle: for a generic point; Right: for a boundary point. Note that for $\alpha \leq 1$, the interpolation only includes immediate neighbors for the octree, reducing the amount of computation.

5 Applications

The continuum of levels of detail we have defined is a powerful general purpose tool. We next present three applications: a novel hierarchical approach to surface extraction using deformable models, a projection-based surface reconstruction algorithm, and ray-tracing of huge point sets.

In what follows, all results are reported on a Pentium IV 2.8 GHz. with 1GB RAM and an IDE hard drive. We used gcc under MS Windows ©. In Figures 1 and 6, we used the algorithm described in [BCS01] to compute per-vertex illumination for our reconstructed meshes.

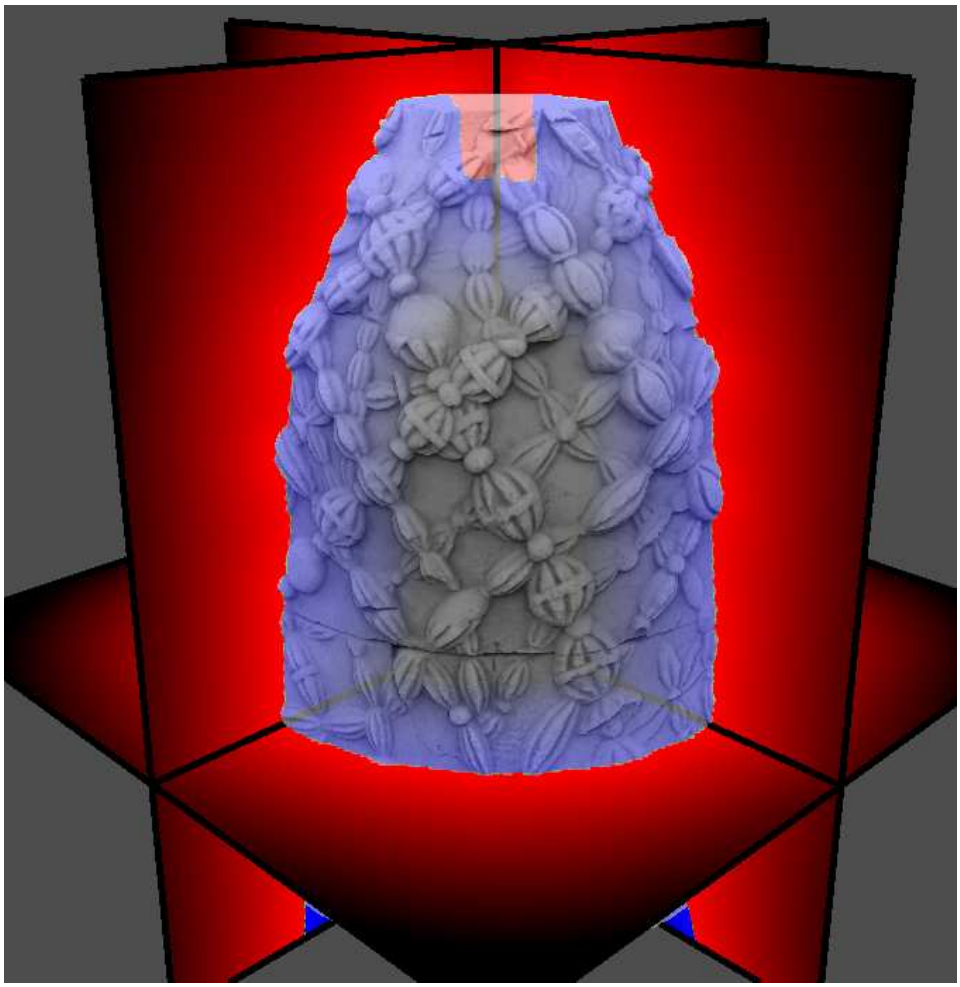

Figure 5: Continuous distance function: results of the volumetric approximation scheme applied on the discrete distance shown in Figure 3 (Omphalos data, octree level 10).

5.1 Hierarchical Snakes

Deformable models offer a well-known framework to extract and optimize a surface under several kinds of constraints. Two main techniques exist: parametric deformable models, such as snakes [KWT88] or balloons [Coh91], and implicit deformable models such as level-sets [CCCD93], [MJV95]. We present in this section an implementation of the classic 3D *snake* technique [CCA92] using a mesh representation of the surface.

Object (*iter*)	*L=6* (size)	*L=7* (size)	*L=8* (size)	*L=9* (size)	*L=10* (size)	*L=11* (size)
St.Matthew (2)	2.1 (2.6k)	6.0 (10.7k)	24 (43k)	1:49 (174k)	8:10 (700k)	27:40 (2.5M)
Armadillo (5)	4.0 (3.4k)	7.8 (3.8k)	17 (15.4k)	1:01 (63k)	4:17 (256k)	(n.a.)

Table 2: Timings of the snake evolution computations. *iter* is the number of inner snake iterations per 1/10th level steps. *time* is the processing time in seconds per thousands of vertices for a complete step with *iter* snake iterations. *L=6..10* is the accumulated time (min:sec) for the snake evolution including initialization at level 5. The size (in parenthesis) represents the number of triangles in the resulting mesh.

The deformable model framework allows us to formulate our problem as a global energy optimisation problem. After developing the corresponding Euler-Lagrange equation, the final snake surface S is found as the solution of the following iterative equation:

$$\frac{\partial S}{\partial t} = \mathcal{F}_{ext}(S) + \mathcal{F}_{int}(S), \tag{14}$$

the discrete version being:

$$S_{t+1} = S_t + \Delta t(\mathcal{F}_{ext}(S_t) + \mathcal{F}_{int}(S_t)). \tag{15}$$

The equation is composed of two terms: an external data-driven term $\mathcal{F}_{ext}(S)$ and an internal regularisation term $\mathcal{F}_{int}(S)$. For our particular problem, the data term is simply defined as $\mathcal{F}_{ext} = -d\mathbf{u}$, where d is our smooth hierarchical distance and $\mathbf{u}$ its associated guidance vector field. Concerning the regularisation term $\mathcal{F}_{int}$, we use a linear combination of a Laplacian regularisation term, that controls the tension of the model, and a biharmonic regularisation term, that controls its rigidity. As discussed in [DMSB99], both operators do not perform well when the edges of the mesh have very different sizes. As a result, an additional remeshing step is done at the end of each iteration in order to maintain a minimum and a maximum distance between neighboring points of the mesh. This is controlled by decimation and refinement of the mesh, based on the edge collapse operator and the $\sqrt{3}$-subdivision algorithm [Kob00], respectively.

In order to guarantee appropriate evolution of the snake, the sum of the forces are bounded by a fraction of the snakes minimal edge length. The convergence speed of the snake is thus related to its complexity: more steps are required when we have more vertices. We use our continuum of distance functions to make this algorithm hierarchical. We start with a raw snake with few polygons (a hundred), and make it evolve (Eq. 15) in a coarse yet continuous distance function (e.g. d_5). Then, we progressively increase the level λ of the distance function d_λ together with the complexity of the snake (induced by the edge length criterion), and iterate. If the level steps $\Delta\lambda$ are small enough, very few iterations per level step are necessary for convergence. In the examples we have tested, 2 to 5 iterations proved to be sufficient for a level step $\Delta\lambda = 1/10$. The snake finally converges to the isosurface at maximal level k_{max}, with a small number of iterations and with increasing number of

polygons. Timings are shown in Table 2, and resulting meshes in Figure. 1. We can see that the generation of the snake for distance level 9 is less than two minutes, and its visual quality is already high. Generating the same mesh with previous approaches [XP98, HS04] required up to two hours, using the St.Matthew at level 9, where our approach converges in less than two minutes.

However, this method has two limitations: it cannot easily handle changes in topology, and has high memory requirements. Since it makes extensive use of connectivity and performs remeshing operations, the mesh data structure needs to be stored in memory in its entirety throughout the process. This limits the size of the models which can be generated. We thus present in the next section a dual contouring technique driven by our hierarchical distance function. This technique can capture topology and can generate larger meshes. Although very fast for low resolution, the contouring approach is much slower than snakes for higher resolution. However the two techniques can be combined. If the topology of the object we want to reconstruct can be captured with a coarse grid (a low resolution mesh), the result of the next method can be used as input for the hierarchical snake approach.

5.2 Distance-driven Dual Contouring

The alternate approach we propose for surface reconstruction (meshing) of any isosurface of our distance function can be applied on any level of detail. We base our algorithm on the ideas of Surface Nets and Dual Contouring presented in [JLSW02]. The main difference is that we have a projection operator to compute a suitable vertex position, while in [JLSW02] a quadric error function is minimized using SVD.

We cast rays in a grid-like sample pattern of resolution *grid-res*: for each grid edge we cast a ray following the edge line and we gather each intersection with the object. We mark each grid edge that contains an intersection and that will correspond to a quad in the final mesh. Then, for each of the four cells connected to a marked edge, we mark the cell as containing a vertex. The vertex $\mathbf{v}$ is initially placed at the center of the cell. A face is built by connecting the four vertices. Vertices are then moved towards the surface using a simple projection operator Π:

$$\Pi(\mathbf{v}) = \mathbf{v} - d(\mathbf{v})\mathbf{u}$$

This projection operator can be applied a few times to converge more precisely to the close surface.

Using this meshing scheme, we only store the grids cells with associated vertices in an acceleration structure. Then each insertion of a vertex in a grid cell is in fact a query for a new or an existing vertex index. This does not require much storage, and very large meshes can thus be constructed efficiently. These meshes are by construction manifold and hole-free. Timings and memory are shown in Table 3.

In order to avoid self-intersecting meshes (due to the projection operator), it is preferable to use a grid with resolution at least as high as the octree of the level used to evaluate the distance.

Model	*grid-res*	*cast*	*project*	*size*	ram
St.Matthew	64^3	5.2	0.16	6.0 k	2.2
Armadillo	256^3	2:03	5.23	163 k	30
H. Buddha	512^3	9:55	15.9	535 k	89
Omphalos	512^3	16:28	0:44	1,006 k	192
Omphalos	1024^3	1h20:23	3:01	4,349 k	666
St.Matthew	1024^3	34:11	0:55	1,658 k	182
St.Matthew	2048^3	2h48:50	4:34	6,788 k	693

Table 3: Timings for reconstruction using the proposed distance-driven dual contouring algorithm. *cast* corresponds to the computation of faces (edge casting) and for the vertices insertion; *project* corresponds to the vertex projection on the isosurface; and *size* is the number of vertices in the resulting mesh. *ram* is the peak memory usage, in megabytes.

5.3 Iso-surface Ray Tracing

Ray-tracing of various isodistances (to shrink or fatten the object) and at different levels of detail can be performed easily using our structure. We ray-trace the hierarchical distance function and get the position on the ray where we have the desired offset value μ, working at a given level of detail λ. We use a recursive algorithm for the intersection between a ray and an isosurface $\mathcal{S}(\lambda, \mu) = \{\mathbf{x} : d_\lambda = \mu\}$. First, we get the two intersection points (A and B) of the ray with the domain box of our distance function and we evaluate the distance at the two points. We then use a splitting criterion: if both points are farther than $\kappa|AB|$ from the object, there is no intersection and we discard the segment. κ is a coefficient to account for the fact that our distance function is not the Euclidian distance to the object. A κ value of 1.1 provided excellent results with no inconsistencies. If each point has a different distance sign, we know we have an intersection. Otherwise, the result is unknown. If we have an intersection or we are in the unknown situation, we split the segment into two sub-segments, and iterate until a size threshold for the segment has been reached. Below this threshold, the unknown result is discarded, and the intersection case results in an intersection point M interpolated from the two distance values. The guidance vector $\mathbf{u}$ is then interpolated and used as an approximation of the surface normal. Results of ray-tracing are shown in Figure 8.

6 Discussion

We have presented an algorithm to generate a hierarchical distance function from points and normals. This distance function is continuous, with a continuous gradient, allowing the definition of a continuum of levels of detail. Three applications of our approach have been presented: a fast hierarchical surface reconstruction algorithm using deformable models, a distance-driven dual contouring algorithm, and ray-tracing of huge point sets. Since our

distance function is signed, any hole or undersampling issue in the input data is automatically corrected. Reconstructed meshes are thus hole-free.

Our method can compare to MPU [OBA+03] since we also build an octree, which stores information for an implicit function reconstruction. Our method differs mainly in two points. First, we require a working level k_{max} which controls the size of the structure, where MPU requires a precision parameter for the implicit function. On the other hand, MPU has an inherent simplification step for the octree in a way similar to [FPRJ00]. Second, we build a continuum of levels of detail in one pass, where MPU requires one pass for each level of detail. Our total computation time is comparable to the computation time for the MPU algorithm for a single error threshold.

In the initialization phase, we average the set of points and normals belonging to the same leaf cells, which is prone to error if many points are in the same cell. However we have not encountered that problem even with the St.Matthew data. Since we consider input data as ground truth for the surface definition, if error is present, such noise should be treated beforehand. For example, in the case of range data misregistration, the signed distance cannot be consistently defined by the input.

As future work, we want to handle geometric features in the hierarchical distance definition, in order to represent sharp edges and corners in the resulting zero isosurface at different levels of detail. We would also like to handle other attributes such as color or appearance. Other applications of our hierarchical distance function are possible. For example, the guidance vector field and the resulting projection operator could be used to render Hypertextures as in [PH89]. This hierarchical distance function could also be used for distant object interaction, up to collision detection, continuously.

Figure 6: Results of distance-driven surface reconstruction on the Omphalos model (4 million vertices, level 10 of the octree, grid resolution: 1024^3).

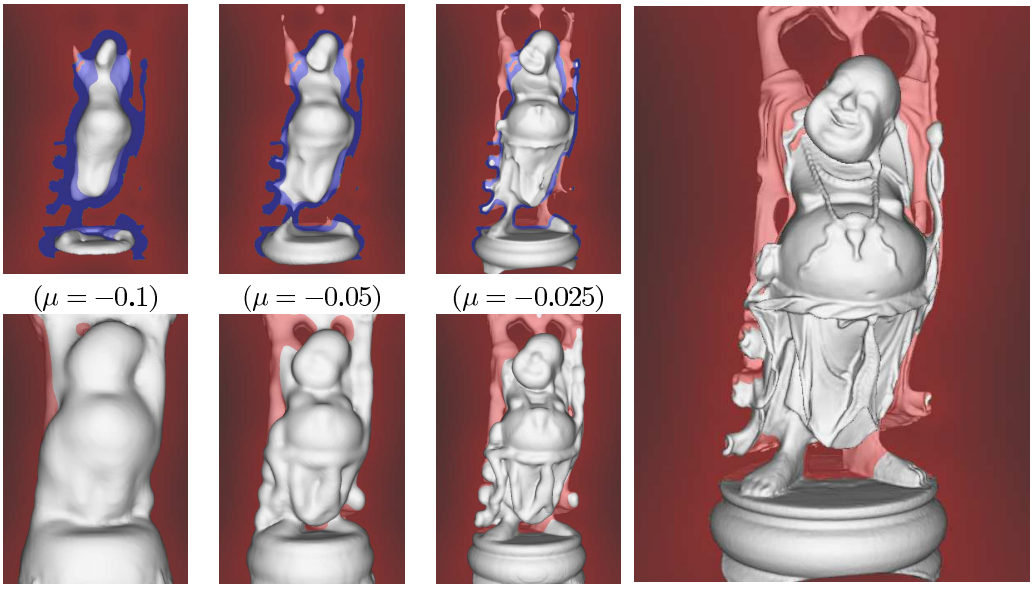

Figure 7: Iso-surfaces of the Happy Buddha data reconstructed using the distance-driven dual contouring algorithm. The model is rendered with a semi-transparent cut of the distance field. Note the topology changes and the smoothness of the isosurface for the various offsets μ.

Figure 8: The dancers model: 49,500,000 input points level 11 of the octree, and raytraced at image resolution 1670x3400. Rendering this image required about 4 hours (two light sources, and maximal reflexion depth of 2.)

References

[AK04] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. *ACM Trans. Graph.*, 23(3):264–270, 2004.

[Bae01] J. Andreas Baerentzen. On the implementation of fast marching methods. Technical Report IMM-REP-2001-13, Technical University of Denmark, 2001.

[BCS01] R. Borgo, P. Cignoni, and R. Scopigno. An easy to use visualization system for huge cultural heritage meshes. In *Proceedings of the VAST'01 Conference*. Eurographics, November 2001.

[Blo94] Jules Bloomenthal. *An Implicit Surface Polygonizer*, chapter IV.8. AP Professional (Academic Press), 1994.

[CBC+01] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM Press, 2001.

[CCA92] Isaac Cohen, Laurent D. Cohen, and Nicholas Ayache. Using deformable surfaces to segment 3-D images and infer differential structures. *CVGIP*, 56(2):242–263, September 1992.

[CCCD93] V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours. *Numerische Mathematik*, 66:1–31, 1993.

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH 1996*. ACM, ACM Press / ACM SIGGRAPH, 1996.

[Coh91] L. D. Cohen. On active contour models and balloons. *CVGIP: Graphical models and Image Processing*, 53(2):211–218, March 1991.

[DMGL02] James Davis, Steven R. Marshner, Matt Garr, and Marc Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Symposium on 3D Data Processing, Visualization and Transmission*. IEEE, June 2002.

[DMSB99] Mathieu Desbrun, Mark Meyer, Peter Schroder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH'99*, pages 317–324. ACM, 1999.

[FPRJ00] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: A general representation of shape for computer graphics. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, pages 249–254, New York, 2000. ACM, ACM Press / ACM SIGGRAPH.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 2000 Proceedings*, pages 209–216. ACM, 1997.

[Gib98] Sarah F. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *IEEE Symposium on Volume Visualization*, pages 23–30, 1998.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John MacDonald, and Werner Stuetzle. Mesh optimization. In *SIGGRAPH'93 Proceedings*, pages 19–26. ACM, August 1993.

[HHVW96] Taosong He, Lichan Hong, Amitabh Varshney, and Sidney Wang. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):171–184, 1996.

[HLC+] Jian Huang, Yan Li, Roger Crawfis, Shao-Chiung Lu, and Shuh-Yuan Liou. A complete distance field representation.

[Hop96] Hugues Hoppe. Progressive meshes. In *SIGGRAPH'96 Proceedings*, pages 99–108. ACM, August 1996.

[HS04] Carlos Hernández and Francis Schmitt. Silhoutte and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding, Special issue on "Model-based and image-based 3D Scene Representation for Interactive Visualization"*, 96(3):367–392, december 2004.

[JG96] Derek Jung and Kamal K. Gupta. Octree-based hierarchical distance maps for collision detection. *Journal of Robotic Systems*, June 1996.

[JLSW02] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, July 2002.

[Kob00] L. Kobbelt. $\sqrt{3}$-subdivision. In *SIGGRAPH 2000*, pages 103–112. ACM, 2000.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–332, 1988.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM, ACM Press, 1987.

[LGF04] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. *ACM Trans. Graph.*, 23(3):457–462, 2004.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project. In Kurt Akeley, editor, *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 131–144, New York, 2000. ACM, ACM Press / ACM SIGGRAPH.

[LPZ03] S. Leopoldseder, H. Pottman, and H. Zhao. The d2-tree: A hierarchical representation of the squared distance function. Technical report, Institute of Geometry, Vienna Institute of Technology, 2003.

[MJV95] R. Malladi, J.A.Sethian, and B.C. Vemuri. Shape modelling with front propagation: A level set approach. *IEEE Tr. on PAMI*, 17(2):158–175, February 1995.

[OBA+03] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. *ACM Transactions on Graphics*, 22(3):463–470, July 2003.

[PH89] K. Perlin and E. M. Hoffert. Hypertexture. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 253–262. ACM, ACM Press, 1989.

[Pop03] Stéphane Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Computational Physics*, (190):572–600, July 2003.

[SFYC96] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Frederic Cronhill. Octree-based decimation of marching cube surfaces. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 335–ff. IEEE, IEEE Computer Society Press, 1996.

[SOM04] Avneesh Sud, Miguel A. Otaduy, and Dinesh Manocha. Difi: Fast 3d distance field computation using graphics hardware. In Marie-Paule Cani and M. Slater, editors, *Proceedings of EUROGRAPHICS 2004*, volume 23,(3). Eurographics, 2004.

[SOS04] Chen Shen, James F. O'Brien, and Jonathan R. Shewchuck. Interpolating and approximating implicit surfaces from polygonal soups. *ACM Transactions on Graphics*, 23(3):896–904, August 2004.

[TO02] Greg Turk and James F. O'Brien. Modelling with implicit surfaces that interpolate. *ACM Trans. Graph.*, 21(4):855–873, 2002.

[VCBS03] Joan Verdera, Vicent Caselles, Marcelo Bertalmio, and Guillermo Sapiro. Inpainting surface holes. In *Proceedings of IEEE Internaltional Conference on Image Processing*. IEEE, September 2003.

[XP98] Chenyang Xu and Jerry L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3), March 1998.

[ZOF01] Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. Fast surface reconstruction using the level set method. In *VLSM'01 : Proceedings of the IEEE Workshop on Variational and Level Set Methods*, pages 194–202. IEEE, 2001.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399