J. KSIAM Vol.16, No.1, 31–49, 2012

SURFACE RECONSTRUCTION FROM SCATTERED POINT DATA ON OCTREE

CHANGSOO PARK[1], CHOHONG MIN[2], AND MYUNGJOO KANG[1,†]

[1]DEPARTMENT OF MATHEMATICAL SCIENCES, SEOUL NATIONAL UNIVERSITY, SOUTH KOREA
E-mail address: {winspark,mkang}@snu.ac.kr

[2]DEPARTMENT OF MATHEMATICS, EWHA WOMANS UNIVERSITY, SOUTH KOREA
E-mail address: chohong@ewha.ac.kr

ABSTRACT. In this paper, we propose a very efficient method which reconstructs the high resolution surface from a set of unorganized points. Our method is based on the level set method using adaptive octree. We start with the surface reconstruction model proposed in [20]. In [20], they introduced a very fast and efficient method which is different from the previous methods using the level set method. Most existing methods[21, 22] employed the time evolving process from an initial surface to point cloud. But in [20], they considered the surface reconstruction process as an elliptic problem in the narrow band including point cloud. So they could obtain very speedy method because they didn't have to limit the time evolution step by the finite speed of propagation.

However, they implemented that model just on the uniform grid. So they still have the weakness that it needs so much memories because of being fulfilled only on the uniform grid. Their algorithm basically solves a large linear system of which size is the same as the number of the grid in a narrow band. Besides, it is not easy to make the width of band narrow enough since the decision of band width depends on the distribution of point data. After all, as far as it is implemented on the uniform grid, it is almost impossible to generate the surface on the high resolution because the memory requirement increases geometrically.

We resolve it by adapting octree data structure[12, 11] to our problem and by introducing a new redistancing algorithm which is different from the existing one[19].

1. INTRODUCTION

Surface reconstruction from a set of unorganized points also known as reverse engineering is a very popular and challenging problem in many area such as medical image, computer graphics, mechanical engineering. Since problem to find the connection between point data is ill-posed, we can just approximate the underlying surface. Although there are so many methodologies to do this, these methods are broadly classified into two categories: explicit representation and implicit representation. The former consists of Delaunay triangulations,

Received by the editors September 1 2011; Accepted February 10 2012.
2010 *Mathematics Subject Classification.* 65M55, 68U10.
Key words and phrases. Level set method, surface reconstruction, point cloud, octree.
† Corresponding author.

Voronoi diagrams, the power crust[3] and so on. The latter includes Hoppe's methods[8], moving least squares[1, 6], the radial basis function[4], the level set method[21, 22], etc.

The explicit method is the method which finds a graph to connect every pair of point data. The α-shape[5], the β-skeleton[2] and the power crust[3] are examples of explicit representation. These graphs are not only based on the Voronoi diagram and the Delaunay triangulation but also the subsets of the Delaunay triangulation.

The implicit representation uses a scalar function which describe a surface as its zero level set.

The first and most famous algorithm using implicit representation is by Hoppe et al.[8]. It firstly approximates the tangent plane on a point using least squares on k nearest neighboring points. Then it considers the signed distance from the point to its projection onto the tangent plane as the signed distance on the whole domain.

Radial basis function is very well-known function in many fields. In [4], the surface reconstruction using radial basis function was introduced. While it can get the smooth surface, it has some limitations. It should solve the linear system $Ax = b$ which the matrix A is dense and ill-conditioned. Although they employed FMM(Fast Multipole Method) for evaluations of energy minimization, it is rather slow compared with other methods. Also, it needs additional information called *off-surface points* besides given point data.

The first application of the level set method to our problem is the result by Zhao et al. in [21] and [22]. Their models make use of the surface evolving from an closed surface of which inside region includes all point data as the level set method comes originally from computational fluid dynamics. The level set method has the advantages of handling topological changes easily and not requiring the normal information on point data. But the previous models have the bounds of speed from the CFL condition because they use time-dependent PDEs such as nonlinear parabolic PDE and linear advection equation.

In [20], Ye et al. approached our problem from a different standpoint. They considered the surface reconstruction as a Poisson problem. Because they became free from the CFL condition, they could improve the efficiency of the process. However, their algorithm basically solves a large linear system of which size is the same as the number of the grid in a narrow band and is fulfilled only on the uniform grid. Consequently, it requires so much memories. Therefore, it is almost impossible to generate the surface on the high resolution because the memory requirement increases geometrically.

In our problem, the part to be described accurately is the surface passing through points i.e. the interface of the level set function rather than the whole domain. Therefore we have only to capture the details near the interface. That is, we need the multi-resolution approach. The adaptive grid can resolve that problem with refining only near the interface. The adaptive grid is very efficient and prevalent in many fields such as computer graphics and scientific computing and computational fluid dynamics where they need to reduce much computational costs. Also in surface reconstruction, there are some researches[13, 17] using the adaptive grid in order to generate high resolution. We employ the octree which is a sort of the adaptive grid. The Poisson's equation can be effectively solved on the octree. There are several Poisson solvers on the octree developed up to date.

In [15], Popinet proposed a second order non-symmetric numerical method to solve the incompressible Euler equations on octree. In his method, the pressure is sampled at the center of each cell and the discretization of the Poisson equation requires interpolation procedures involving the pressure values at several adjacent cells. Consequently, this discretization requires finite difference method to use large support.

In [10], Losasso et al. proposed a first order Poisson solver using octree data structures and applied it to the Navier-Stokes equation. They stored data except the pressure at nodes and the pressure at the center of the cell. They perturbed the location of the pressure by a quantity proportional to the size of a grid cell in order to obtain a symmetric linear system. So most approximations in their scheme is second order accurate but approximations to the gradient of the pressure is first order accurate.

In [11], Min et al. proposed a second order accurate finite difference discretization for the variable coefficient Poisson equation on non-graded grids, which yields second order accuracy covering the solution's gradients. The scheme employs sampling the solution at the nodes of a cell. The discretization at one cell's node only uses nodes of two or three adjacent cells, producing schemes that are straightforward to implement.

Adaptive level set method was successfully developed in Min's paper [11]. Ghost nodes had been treated with linear interpolation. Without increasing the support of numerical methods, they showed a possible quadratic interpolation. We apply it to our problem.

The outline of the paper is as follows. In the next section, we explain mathematical models using level set method. In section 3, we present numerical method to reconstruct a surface from point cloud on the octree. In section 4, we show experimental results to demonstrate that our method is efficient.

2. MATHEMATICAL MODELS

The most well-known variational model of surface reconstruction using the level set method is the weighted minimal surface model[21]. Let $\mathcal{S}$ denote the data set which can include data points, pieces of curves or surfaces. In our problem, $\mathcal{S}$ means data points. Define $d(\mathbf{x}) = \text{dist}(\mathbf{x}, \mathcal{S})$ to be the distance function to $\mathcal{S}$, where $\text{dist}(\mathbf{x}, \mathcal{S}) = \min_{\mathbf{y}\in\mathcal{S}} \parallel \mathbf{x} - \mathbf{y} \parallel_2$. Then this model begins to define the following surface energy:

$$E(\Gamma) = \left[\int_\Gamma d^p(\mathbf{x}) ds \right]^{\frac{1}{p}}, \quad 1 \le p \le \infty, \tag{2.1}$$

where Γ is an arbitrary surface and ds is the surface area. To find the surface of which energy is minimized, they calculate the gradient descent of the functional (2.1).

$$\frac{d\Gamma}{dt} = -\left[\int_\Gamma d^p(\mathbf{x}) ds \right]^{\frac{1}{p}-1} d^{p-1}(\mathbf{x}) \left[\nabla d(\mathbf{x}) \cdot \mathbf{n} + \frac{1}{p} d(\mathbf{x}) \kappa \right] \mathbf{n}, \tag{2.2}$$

where $\mathbf{n}$ is the unit outward normal and κ is the mean curvature. The flow from the above gradient (2.2) rules the movement of the surface. If the surface is initially far from data set, the surface evolves into the final surface with keeping the balance between the potential force

$\nabla d(x) \cdot \mathbf{n}$ and the surface tension $d(\mathbf{x})\kappa$. The scalar function d($\mathbf{x}$) allows the surface to be more flexible in regions close to the data set and to be more rigid in regions distant from the data set.

Here level set method is used to handle topological changes and to obtain an implicit surface. Let $\Omega(t)$ be the (generally multiply connected) region enclosed by $\Gamma(t)$. Let $\phi(\mathbf{x}, t)$ be the level set function associated with $\Omega(t)$; i.e.,

$$\begin{array}{lll} \phi(\mathbf{x}, t) < 0 & \text{in} & \Omega(t), \\ \phi(\mathbf{x}, t) = 0 & \text{on} & \Gamma(t), \\ \phi(\mathbf{x}, t) > 0 & \text{in} & \bar{\Omega}^c(t). \end{array}$$

Then $\Gamma(t)$ is identical to the zero level set of $\phi(\mathbf{x}, t)$. As [21], level set formulation of (2.1) is

$$E(\Gamma) = \left[\int_\Gamma d^p(\mathbf{x}) ds \right]^{\frac{1}{p}} = E(\phi) = \left[\int d^p(\mathbf{x}) \delta(\phi(\mathbf{x})) |\nabla \phi(\mathbf{x})| d\mathbf{x} \right]^{\frac{1}{p}}, \tag{2.3}$$

where $\delta(\mathbf{x})$ is the one-dimensional delta function and $\delta(\phi(\mathbf{x}))|\nabla\phi(\mathbf{x})|d\mathbf{x}$ is the surface area element at the zero level set of ϕ. Also the gradient flow for ϕ corresponding to (2.2) is

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| \left[\int d^p(\mathbf{x}) \delta(\phi) |\nabla \phi| d\mathbf{x} \right]^{\frac{1}{p}-1} d^{p-1}(\mathbf{x}) \left[\nabla d(\mathbf{x}) \cdot \frac{\nabla \phi}{|\nabla \phi|} + \frac{1}{p} d(\mathbf{x}) \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right], \tag{2.4}$$

where $\frac{\nabla \phi}{|\nabla \phi|}$ and $\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$ are the level set representation of the unit normal and the mean curvature respectively.

In [22], Zhao et al. proposed the following convection model which is similar to the previous minimal surface model in methodology but uses the different physical model. The convection of a flexible surface Γ in a velocity field $v(\mathbf{x})$ is described by

$$\frac{d\Gamma(t)}{dt} = v(\Gamma(t)). \tag{2.5}$$

If the velocity field is created by a potential field $\mathcal{F}$, then $v = -\nabla \mathcal{F}$. Because the distance function $d(\mathbf{x})$ to the data set $\mathcal{S}$ means the potential field in this convection model, the convection equation can be represented by

$$\frac{d\Gamma(t)}{dt} = -\nabla d(\mathbf{x}). \tag{2.6}$$

The level set formulation of the equation (2.6) is

$$\frac{\partial \phi}{\partial t} = \nabla d(\mathbf{x}) \cdot \nabla \phi. \tag{2.7}$$

The evolution equation (2.2) is a nonlinear parabolic equation because it has the mean curvature of the surface. Since the convection equation (2.5) is a first order linear differential equation which has a time step $\Delta t = O(h)$ where h is the grid size, it saves the time over parabolic $\Delta t = O(h^2)$ time step restrictions.

The above two models handle the surface reconstruction problem as the time evolution equation. In [20], Ye et al. considered our problem as Poisson's equation. They introduced the

following energy functional of ϕ :
Given point data $\{\mathbf{x}_l\}_{l=1,\dots,N} \subset \mathcal{S}$,

$$E(\phi) = \int G(\phi(\mathbf{x}))d\mathbf{x} + \sum_{l=1}^{J} \beta_l (P_l\phi)^2, \tag{2.8}$$

where the projection operator $P_l\phi = \int p_l(\mathbf{x})\phi(\mathbf{x})d\mathbf{x}$ and $\int p_l(\mathbf{x})d\mathbf{x} = 1$.
In case the data is uniformly distributed, they set $G(\phi(\mathbf{x})) = |\nabla\phi(\mathbf{x})|^2$. Then the first term play a role as the diffusion term which determines the smoothness of the surface and the second term functions as the fidelity term which fits the surface to point data. In the second term the weight parameter β_l affects the accuracy of fitting the surface. If enough large β_l is chosen, even crude initial boundary conditions result in very successful fitting.

Without getting the Euler-Lagrange equation from the energy functional (2.8) directly, they addressed the equation discretized from (2.8). In the case of two-dimensional, the equation is as follows.

$$\bar{E}(\phi) = \sum_i \sum_j \Big(\frac{\phi_{i+1,j} - \phi_{i,j}}{h}\Big)^2 + \Big(\frac{\phi_{i,j+1} - \phi_{i,j}}{h}\Big)^2 + \sum_l \beta_l [\bar{P}_l\phi]^2, \tag{2.9}$$

where $\bar{P}_l\phi = \sum_{k,n} \bar{p}^l_{k,n}\phi_{k,n}$ and $\sum_{k,n} \bar{p}^l_{k,n} = 1$.
Differentiating the energy functional (2.9) with respect to $\phi(i,j)$ at the grid point (i,j), they obtain the Euler-Lagrange equation as follows

$$\frac{1}{2}\frac{\delta\bar{E}}{\delta\phi_{i,j}} = -\frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}}{h^2} + \sum_{l=1}^{J} \beta_l p^l_{i,j} P_l\phi = 0, \tag{2.10}$$

where J is the total number of neighboring points of grid point (i,j), operator P_l is a bilinear interpolation operator and $P_l\phi$ represents the interpolated value of function ϕ at a point $\mathbf{x}_l$.

$$P_l\phi = p^l_{i,j}\phi_{i,j} + p^l_{i,j+1}\phi_{i,j+1} + p^l_{i+1,j}\phi_{i+1,j} + p^l_{i+1,j+1}\phi_{i+1,j+1}, \tag{2.11}$$

where $p^l_{i,j} = \dfrac{(h-r_1)(h-r_2)}{h^2}$, $p^l_{i,j+1} = \dfrac{(h-r_1)r_2}{h^2}$, $p^l_{i+1,j} = \dfrac{r_1(h-r_2)}{h^2}$,
$p^l_{i+1,j+1} = \dfrac{r_1 r_2}{h^2}$.

In three dimension, this model can be extended using trilinear interpolation operator instead of bilinear operator.

After they set initial boundary conditions in the narrow band with tagging algorithm such as the Breadth-First Search, they approximated the underlying surface fitting point data by solving the Poisson's equation (2.10).

This model has the distinct characteristic compared to the previous models as it is a different type of PDE. It doesn't need to be constrained by the CFL condition and its implementation can be improved because there are many efficient algorithms for solving Poisson's equation.

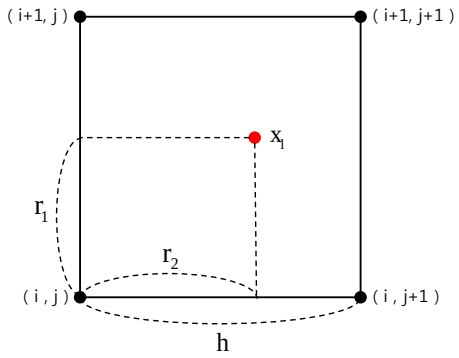

FIGURE 1. The description of 2D interpolation on the uniform grid

However, their method still has the limitation that it needs so much memories because of being fulfilled only on the uniform grid. Their algorithm basically solves a large linear system of which size is the same as the number of the grid in a narrow band. If it is possible to make the width of band narrow enough, a small-sized linear system can be obtained. But in the above model, the decision of band width depends on the distribution of point data, more specifically, the maximum of distances between point data. It means that if point data are ranged sparsely or there is a big hole in the data, the width of band cannot help but become broad. Surely, these cases always exist because most point data from real objects are unorganized. Therefore, as far as it is implemented on the uniform grid, it is almost impossible to generate the surface on the high resolution because the memory requirement increases geometrically. So we needs adaptive grid because it result in the linear system of reasonable size even on high resolution grid. We employ data structures of octree to implement multi-resolution adaptively. Our scheme is based on Min's[11].

3. NUMERICAL METHOD

We denote a (rectangular) cell in quadtree or octree by C and the node of cell by v.

3.1. **Tree Generation and Splitting Condition.** We use a standard quadtree(resp. octree) data structure to represent the spatial discretization of the two(resp. three)-dimensional domain. Starting from the root of tree corresponding to the whole domain, each cell is split into four(resp. eight) children until the desired level of detail is achieved. We refer the reader to the books of Samet[16] for more details on quadtree/octree data structures.

In our case, the above process stops after the finest resolution for each cell containing at least a point is fulfilled. Details of the process is as follows.

First, for a point we start to split all children cell Cs from a root cell according to the splitting condition

$$\text{dist}(\mathbf{x}, \partial C) \leq \frac{1}{2}\min\{\text{width of } C, \text{height of } C\}. \tag{3.1}$$

After finishing the splitting for a point, we split cells satisfying the above condition starting from the root cell for another point. Of course during this process, we don't split cells which is already split.

In our problem, we need the graded tree, which limits the difference of level between two adjacent cells to at most one, in order to secure the uniform grid near the interface. Otherwise, incorrect information on the nodes of big cells compared with neighboring cells can corrupt values at not only neighboring nodes but distant nodes.

Lemma 1. *The splitting condition (3.1) ensures the finest resolution in not only the cell containing the point, but also its neighboring cells. (e.g. 5 cells in quadtree or 7 cells in octree)*

Proof. At first, we consider quadtree case. If a point x is inside a cell, it is obvious that the condition (3.1) is satisfied. Thus the finest resolution is guaranteed for the cell containing the point x. We denote the finest cell containing the point x by C_c. For cells not having the point, we can consider four directional neighboring cells of C_c. We call those C_c^N(north),C_c^S(south),C_c^E(east),C_c^W(west), respectively. We have only to show that the levels of C_c^N,C_c^S,C_c^E,C_c^W are equal to the level of C_c.

Consider the parent cell C_p of C_c. Then C_c is one of four quadrants in C_p. In case that C_c is the first quadrant(i.e. north-east quadrant) in C_p, The north neighboring cell C_p^N and the east neighboring cell C_p^E of C_p satisfy the condition (3.1). Therefore, there are C_c^N and C_c^E of which level equal the level of C_c. Moreover, since C_c^S and C_c^W are children of C_p, the levels of C_c^S and C_c^W are evidently the same as the level of C_c. For the other quadrants, we can get the same result by symmetry.

We also can extend the above discussion to octree easily.

□

3.2. **Distance Function.** After the generation of octree, we calculate the distance from point data $\mathbf{x}$ at each node $\mathbf{v}_i$. This process progresses in a similar way as generating octree but with a condition different from the splitting condition.

To begin with, for a point $\mathbf{x}_1$ chosen arbitrarily, we calculate the distances at nodes of the root cell. Then we check the following condition for each children of the root cell.

$$\text{dist}(\mathbf{x}, \partial C) \leq \max\{\sqrt{(2l_1)^2 + l_2^2}, \sqrt{l_1^2 + (2l_2)^2}\} + \min\{l_1, l_2\}, \tag{3.2}$$

where l_1 is a half of the width of C and l_2 is a half of the height of C. If a cell satisfies the condition (3.2) and distance values at nodes of the cell are less than the existing ones, the distances at nodes of the cell are updated. This job continues until it reaches the level of the finest cell containing the point $\mathbf{x}_1$ along the hierarchy of the octree. Then with another point

$\mathbf{x}_2$, we again calculate the distances at nodes of the cells satisfying the above condition (3.2). It stops after it is carried out for all points $\{\mathbf{x}\}$.

In the process of updating the distances, it is better to search as many nodes as possible because there are some nodes of which distances are not minimum. Of course, if we choose enough large region, updating process will guarantee the minimum distances at all nodes. But distancing gets too time-consuming chore. Thus we need the least area guaranteeing a complete distance function in the entire domain. The condition (3.2) guarantees it.

The distancing on the octree is more efficient compared to that on the uniform grid. Denote the one side resolution of the uniform grid discretized in the entire domain by N and the number of point data by L. On the uniform grid, the time required to obtain the distance function in the whole domain is $O(N^3) + L$. This time came from the fast sweeping algorithm in [23] which is the most efficient algorithm on the uniform grid as far as we know. Meanwhile, we achieved the time of $O(L\log(N))$ in the octree. We can usually regard $L \approx N^2$ because L is the number of points on the interface in some domain. Then the time of distancing on the uniform grid is approximately $O(LN)$ while $O(L\log(N))$ in octree. We will demonstrate this results in section 4.

3.3. **Initial Guess of Signed Distance Function.** We need to initialize boundary conditions in order to solve the Poisson's equation. In addition, boundary condition should contain the information about whether nodes are inside or outside the underlying surface of point data because we want to represent the surface by the zero level set of the level set function ϕ. To achieve it, we employ the fast tagging method in [22] in order to set boundary conditions. Here, we consider just two-dimensional case because it is easy to extend the process to three-dimension. In a way similar to [20], we start from a node $\mathbf{v}_1$ on the boundary of the entire domain to mark the outside region. The node $\mathbf{v}_1$ is unconditionally in the outside region. We check if the distances at neighboring nodes $\{\mathbf{v}_{N_i}\}_{N_i=1,..,8}$ of $\mathbf{v}_1$ satisfies the condition $d(\mathbf{v}_{N_i}) > \epsilon$ where ϵ is a bandwidth-related constant. If there is a node satisfying this condition, we denote the node by the outside. After searching all neighboring nodes of $\mathbf{v}_1$, we pass another node $\mathbf{v}_2$ and do the same job again. When finishing this process for all nodes, we can get the outside region. Then, we set the distances for the outside region to ϵ and the distances for $\{\mathbf{x} : d(\mathbf{x}) < \epsilon\}$ to 0. Finally, the remaining nodes becomes the inside region and the distances for it is set to $-\epsilon$.

The ϵ in the above algorithm is required to satisfy $\epsilon > \frac{1}{2}\epsilon_{data}$, where $\epsilon_{data} = \max_i\{\min_{i\neq j}|x_i - x_j|\}$, to ensure that the region $\{\mathbf{x} : d(\mathbf{x}) > \epsilon\}$ has two topological components. The above choice has some problems. In some data sets, i.e. a set having a large hole, the ϵ_{data} may be large. Then this initial guess can be very rough because of large bandwidth. In this case, our method cannot guarantee appropriate result.

3.4. **Basic Finite Difference Methods on Octree.** In the case of non uniform Cartesian grids, the main difficulty comes from deriving discretizations at T-junction nodes, i.e. nodes for which there is a missing neighboring node in one of the Cartesian directions. For example, Figure 2 depicts a T-junction node $\mathbf{v}_0$, with three neighboring nodes $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ aligned in the Cartesian directions and one ghost neighboring node v_4 replacing the missing grid node in

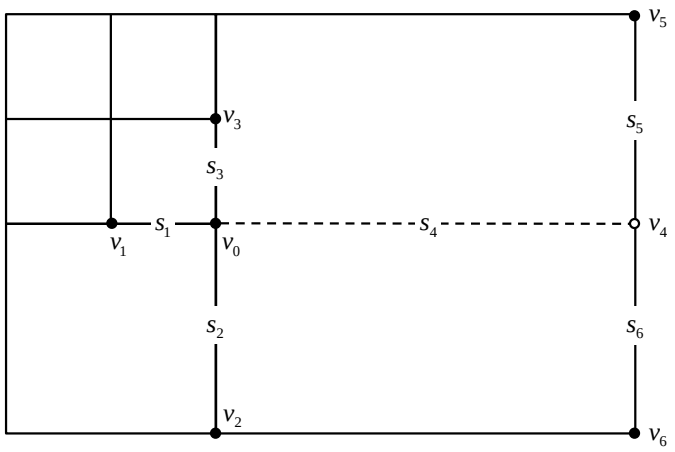

FIGURE 2. Neighboring nodes of a T-junction node, v_0.

the positive Cartesian direction. The value of a node-sampled function $\phi : \{\mathbf{v}_i\} \to \mathbb{R}$ at the ghost node $\mathbf{v}_4$ could for example be defined by linear interpolation:

$$\phi_4^G = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6}. \tag{3.3}$$

However, instead of using this second order accurate interpolation, one can use the following third order accurate interpolation: First, note that a simple Taylor expansion demonstrates that the interpolation error in equation (3.3) is given by:

$$\phi_4^G = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6} = \phi(\mathbf{v}_4) + \frac{s_5 s_6}{2}\phi_{yy}(\mathbf{v}_0) + O(\triangle x_{\text{smallest}})^3, \tag{3.4}$$

where $\triangle x_{\text{smallest}}$ is the size of the smallest grid cell with vertex $\mathbf{v}_0$. The term $\phi_{yy}(\mathbf{v}_0)$ can be approximated using the standard first order accurate discretization $\frac{2}{s_2+s_3}\left(\frac{\phi_2-\phi_0}{s_2} + \frac{\phi_3-\phi_0}{s_3}\right)$ and canceled out in equation (3.4) to give:

$$\phi_4^G = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6} - \frac{s_5 s_6}{s_2 + s_3}\left(\frac{\phi_2 - \phi_0}{s_2} + \frac{\phi_3 - \phi_0}{s_3}\right). \tag{3.5}$$

We also point out that this interpolation only uses the node values of the cells adjacent to $\mathbf{v}_0$, which is particularly beneficial since access to cells not immediately adjacent to the current cell is more difficult and could add on CPU time and/or memory requirement.

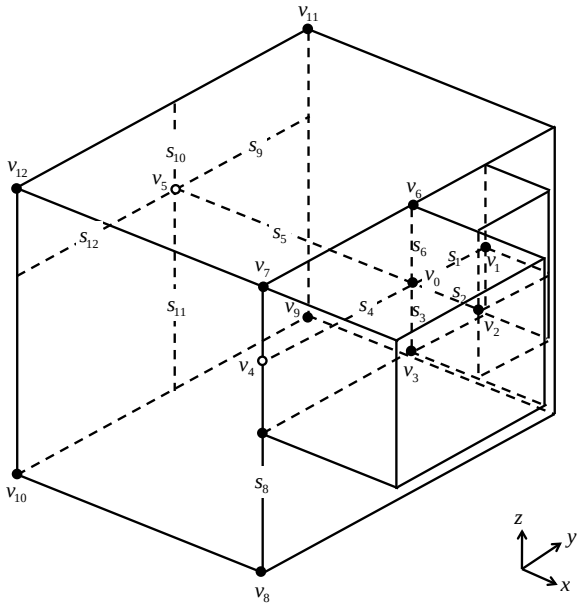

FIGURE 3. Neighboring vertices of a vertex in three spatial dimensions.

In three spatial dimensions, similar interpolation procedures can be used to define the value of ϕ at ghost nodes. Referring to Figure 3, a T-junction node $\mathbf{v}_0$ has four regular neighboring nodes and two ghost nodes. The values of a node-sampled function $\phi : \{\mathbf{v}_i\} \to \mathbb{R}$ at the ghost nodes $\mathbf{v}_4$ and $\mathbf{v}_5$ can be defined by second order linear and bilinear interpolations as:

$$\begin{aligned} \phi_4^G &= \frac{s_7\phi_8 + s_8\phi_7}{s_7 + s_8}, \\ \phi_5^G &= \frac{s_{11}s_{12}\phi_{11} + s_{11}s_9\phi_{12} + s_{10}s_{12}\phi_9 + s_{10}s_9\phi_{10}}{(s_{10} + s_{11})(s_9 + s_{12})}. \end{aligned} \tag{3.6}$$

As in the case of quadtrees, third order accurate interpolations can be derived by canceling out the second order derivatives in the error term to arrive at:

$$\begin{aligned} \phi_4^G &= \frac{s_7\phi_8 + s_8\phi_7}{s_7 + s_8} - \frac{s_7 s_8}{s_3 + s_6}\left(\frac{\phi_3 - \phi_0}{s_3} + \frac{\phi_6 - \phi_0}{s_6}\right), \\ \phi_5^G &= \frac{s_{11}s_{12}\phi_{11} + s_{11}s_9\phi_{12} + s_{10}s_{12}\phi_9 + s_{10}s_9\phi_{10}}{(s_{10} + s_{11})(s_9 + s_{12})} \\ &\quad - \frac{s_{10}s_{11}}{s_3 + s_6}\left(\frac{\phi_3 - \phi_0}{s_3} + \frac{\phi_6 - \phi_0}{s_6}\right) \\ &\quad - \frac{s_9 s_{12}}{s_1 + s_4}\left(\frac{\phi_1 - \phi_0}{s_1} + \frac{\phi_4^G - \phi_0}{s_4}\right). \end{aligned} \tag{3.7}$$

We emphasize that Figure 3 represents the general configuration of neighboring nodes in the case of an octree as described in Min [12].

The third order interpolations defined above allow us to treat T-junction nodes in a same fashion as a regular node, up to third order accuracy. Here, we refer to a regular node as a node for which all the neighboring nodes in the Cartesian directions exist. Therefore, we can then define finite differences for ϕ_x, ϕ_y, ϕ_z, ϕ_{xx}, ϕ_{yy} and ϕ_{zz} at every nodes using standard finite difference formulas in a dimension by dimension framework. For example, referring to Figure 4, we use the standard discretization for ϕ_x and ϕ_{xx}, namely the central difference formulas:

$$\begin{aligned} D_x^0\phi_0 &= \frac{\phi_2-\phi_0}{s_2}\cdot\frac{s_1}{s_1+s_2}+\frac{\phi_0-\phi_1}{s_1}\cdot\frac{s_2}{s_1+s_2}, \\ D_{xx}^0\phi_0 &= \frac{\phi_2-\phi_0}{s_2}\cdot\frac{2}{s_1+s_2}-\frac{\phi_0-\phi_1}{s_1}\cdot\frac{2}{s_1+s_2}, \end{aligned} \tag{3.8}$$

the forward and backward first order accurate approximations of the first order derivatives:

$$\begin{aligned} D_x^+\phi_0 &= \frac{\phi_2-\phi_0}{s_2}, \\ D_x^-\phi_0 &= \frac{\phi_0-\phi_1}{s_1}, \end{aligned} \tag{3.9}$$

and the second order accurate approximations of the first order derivatives:

$$\begin{aligned} D_x^+\phi_0 &= \frac{\phi_2-\phi_0}{s_2}-\frac{s_2}{2}\text{minmod}\left(D_{xx}^0\phi_0, D_{xx}^0\phi_2\right), \\ D_x^-\phi_0 &= \frac{\phi_0-\phi_1}{s_1}+\frac{s_1}{2}\text{minmod}\left(D_{xx}^0\phi_0, D_{xx}^0\phi_1\right), \end{aligned} \tag{3.10}$$

where we use the minmod slope limiter [9, 18] because it produces more stable results in region where ϕ might present kinks. Similarly, approximations for first and second order derivatives are obtained in the y and z directions.

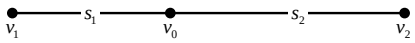

FIGURE 4. One dimensional adaptive grid

3.5. **Numerical Discretization of Model (2.8) on Octree.** We discretize equation (2.8) based on the scheme in the previous section. While the coefficients of the discretized fitting term on octree differs little from that on the regular nodes, the coefficients of the discretized Laplacian term are distinct compared with those on the uniform grid because of T-junction nodes. We need the following process in order to discretize the Laplacian term on octree. Let us consider the conditions of Figure 3. In Figure 3, only $\mathbf{v}_5$ was considered as the ghost node which needs bilinear interpolation. We extend this situation to all neighboring nodes of $\mathbf{v}_0$, that is,

$\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_6$. Then in a similar way as the case of $\mathbf{v}_5$, we can get the values of a node-sampled function at $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{v}_6$ as follows.

$$\begin{aligned} \phi_1^G &= \frac{s_{18}s_{20}\phi_{17} + s_{18}s_{19}\phi_{18} + s_{17}s_{20}\phi_{19} + s_{17}s_{19}\phi_{20}}{(s_{17}+s_{18})(s_{19}+s_{20})} \\ &= \phi(\mathbf{v}_1) + \frac{1}{2}(s_{17}s_{18}\phi_{zz}(\mathbf{v}_0) + s_{19}s_{20}\phi_{xx}(\mathbf{v}_0)) \end{aligned} \tag{3.11}$$

$$\begin{aligned} \phi_2^G &= \frac{s_{13}s_{16}\phi_{13} + s_{14}s_{16}\phi_{14} + s_{13}s_{15}\phi_{15} + s_{14}s_{15}\phi_{16}}{(s_{13}+s_{14})(s_{15}+s_{16})} \\ &= \phi(\mathbf{v}_2) + \frac{1}{2}(s_{15}s_{16}\phi_{zz}(\mathbf{v}_0) + s_{13}s_{14}\phi_{yy}(\mathbf{v}_0)) \end{aligned} \tag{3.12}$$

$$\begin{aligned} \phi_3^G &= \frac{s_{28}s_{29}\phi_{27} + s_{27}s_{29}\phi_{28} + s_{28}s_{30}\phi_{29} + s_{27}s_{30}\phi_{30}}{(s_{27}+s_{28})(s_{29}+s_{30})} \\ &= \phi(\mathbf{v}_3) + \frac{1}{2}(s_{27}s_{28}\phi_{xx}(\mathbf{v}_0) + s_{29}s_{30}\phi_{yy}(\mathbf{v}_0)) \end{aligned} \tag{3.13}$$

$$\begin{aligned} \phi_4^G &= \frac{s_8s_{22}\phi_7 + s_8s_{21}\phi_8 + s_7s_{22}\phi_{21} + s_8s_{21}\phi_{22}}{(s_7+s_8)(s_{21}+s_{22})} \\ &= \phi(\mathbf{v}_4) + \frac{1}{2}(s_7s_8\phi_{zz}(\mathbf{v}_0) + s_{21}s_{22}\phi_{xx}(\mathbf{v}_0)) \end{aligned} \tag{3.14}$$

$$\begin{aligned} \phi_6^G &= \frac{s_{23}s_{26}\phi_{24} + s_{23}s_{25}\phi_{25} + s_{24}s_{26}\phi_{26} + s_{24}s_{25}\phi_{27}}{(s_{23}+s_{24})(s_{25}+s_{26})} \\ &= \phi(\mathbf{v}_6) + \frac{1}{2}(s_{25}s_{26}\phi_{xx}(\mathbf{v}_0) + s_{23}s_{24}\phi_{yy}(\mathbf{v}_0)) \end{aligned} \tag{3.15}$$

Then, we obtain the following equations through simple manipulation.

$$\begin{aligned} \left(\frac{\phi_2^G-\phi_0}{s_2} + \frac{\phi_5^G-\phi_0}{s_5}\right)\frac{2}{s_2+s_5} &= \phi_{xx}(\mathbf{v}_0) + \left(\frac{s_9s_{12}}{(s_2+s_5)s_5} + \frac{s_{13}s_{14}}{(s_2+s_5)s_2}\right)\phi_{yy}(\mathbf{v}_0) \\ &+ \left(\frac{s_{10}s_{11}}{(s_2+s_5)s_5} + \frac{s_{15}s_{16}}{(s_2+s_5)s_2}\right)\phi_{zz}(\mathbf{v}_0) \end{aligned} \tag{3.16}$$

$$\begin{aligned} \left(\frac{\phi_4^G-\phi_0}{s_4} + \frac{\phi_1^G-\phi_0}{s_1}\right)\frac{2}{s_1+s_4} &= \left(\frac{s_{19}s_{20}}{(s_1+s_4)s_1} + \frac{s_{21}s_{22}}{(s_1+s_4)s_4}\right)\phi_{xx}(\mathbf{v}_0) + \phi_{yy}(\mathbf{v}_0) \\ &+ \left(\frac{s_{17}s_{18}}{(s_1+s_4)s_1} + \frac{s_7s_8}{(s_1+s_4)s_4}\right)\phi_{zz}(\mathbf{v}_0) \end{aligned} \tag{3.17}$$

$$\begin{aligned} \left(\frac{\phi_6^G-\phi_0}{s_6} + \frac{\phi_3^G-\phi_0}{s_3}\right)\frac{2}{s_3+s_6} &= \left(\frac{s_{27}s_{28}}{(s_3+s_6)s_3} + \frac{s_{25}s_{26}}{(s_3+s_6)s_6}\right)\phi_{xx}(\mathbf{v}_0) \\ &+ \left(\frac{s_{29}s_{30}}{(s_3+s_6)s_3} + \frac{s_{23}s_{24}}{(s_3+s_6)s_6}\right)\phi_{yy}(\mathbf{v}_0) + \phi_{zz}(\mathbf{v}_0) \end{aligned} \tag{3.18}$$

By multiplying equations (3.16),(3.17),(3.18) by some weights $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$, respectively, we can get a third order accurate approximation of $\Delta\phi$ in three dimensions.

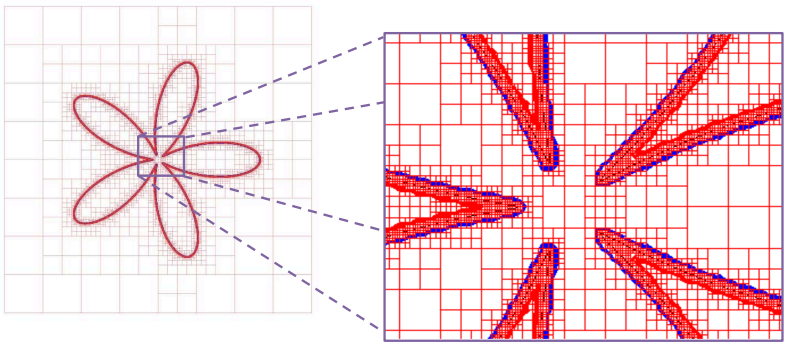

FIGURE 5. The left picture represents the initial narrow band with the boundary condition. The right picture is the magnified concave part of five-leafed clover. The red thick lines denote the outside boundaries and the blue thick line denote the inside boundaries. The black dots represent point data. The red squares show the grid on the octree. The fine grids cluster near the point data while the coarse grids is far from points.

TABLE 1. The processing time and the required memory for reconstructing the two-dimensional five-leafed clover

Resolution	Uniform Grid		Octree	
	Time(sec)	Memory(MB)	Time(sec)	Memory(MB)
512^2	0.15	9.7	0.35	2.6
1024^2	0.52	66.4	0.42	4.5
2048^2	2.05	234.4	0.53	15

4. EXPERIMENTAL RESULTS

We tested our method and Ye's[20] with some two-dimensional and three-dimensional examples. These tests were executed on a PC with 3.16GHz Intel Core 2 Duo CPU and 4GB RAM. We adopted the Biconjugate Gradient Stabilized (BICGSTAB) algorithm to the process of minimizing the functional (2.8). The stopping criterion used is that 2-norm of the residual is less than 10^{-8}.

4.1. **Five-leafed clover.** First, we generated an artificial point data of which shape is five-leafed clover on two-dimension. The data is made up of 5000 uniformly distributed points. We reconstructed the contour from the data by varying the finest resolution from 512^2 to 2048^2.

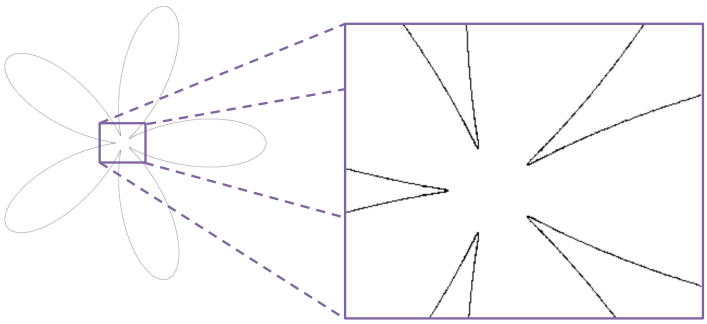

FIGURE 6. The left picture shows the reconstructed five-leafed clover from point cloud. The right picture is the magnified concave part of five-leafed clover. This indicates that the curve is well-fitted to even the structure with high curvature.

Of course, our method can be implemented with quadtree in two-dimension. We compares the processing time and the required memory on quadtree with on the uniform grid in Table 1. Like Table 1 shows, the processing time on quadtree gets shorter than on the uniform grid as the resolution increases. Furthermore, the difference of the memory requirement is much greater than the difference of the time.

As this example was comparatively smooth in shape, the qualities of the curves reconstructed on several different resolutions made no difference. So we show one of those in Figure 6. Figure 5 represents the initial condition for solving the minimization of the functional (2.8), that is, the initial narrow band.

4.2. **Bunny, Dragon, Happy Buddha.** In three dimensional, we reconstructed the Stanford bunny and the Stanford dragon and happy Buddha. Figure 8 and Figure 9 show results on grids of which maximum resolution is 256^3 for the Stanford bunny and the Stanford dragon, respectively. Actually, although we obtained the result of 1024^3 resolution with octree on our PC, the difference of the quality was not noticeable in the case of bunny and dragon. So we demonstrate just 256^3 case among those. But we could confirm the improvement of detail according to an increase in the resolution from the reconstruction for happy Buddha(Figure 10).

In Table 2, the table on the top shows the processing time and the required memory on the uniform grid and the table on the bottom demonstrates the time and the memory on octree. In case the resolution is higher than 256^3, we could not execute the algorithm on the uniform grid in our test environment because of memory depletion. Likewise two-dimensional case, octree

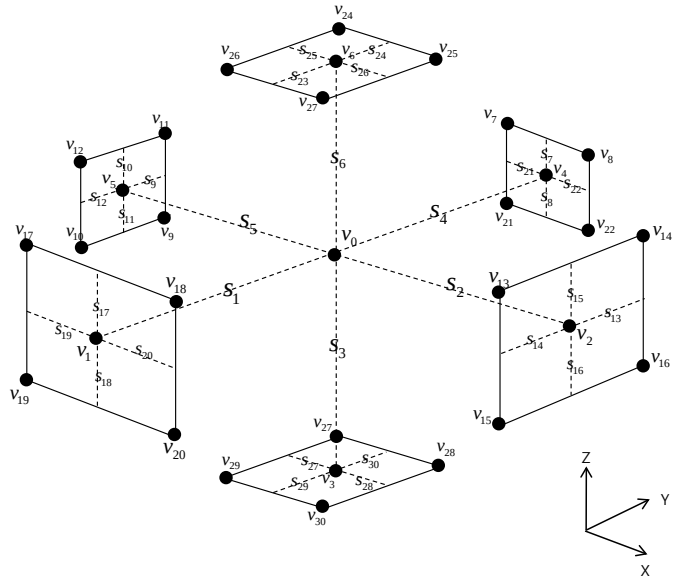

FIGURE 7. We consider all neighboring nodes of a vertex $\mathbf{v}_0$ as the ghost nodes in our scheme. But this situation never happens actually.

structure saved much processing time compared with uniform grid as the resolution get higher. Only in low resolution on octree structure, e.g. 128^3, the time for calculating the unsigned distances is longer than on the uniform grid, in particular, for dragon and Buddha. As we mentioned it in the last paragraph of section 3.2, it is because the distancing on octree depends on the number of point data while the distancing on the uniform grid depends on the number of grids only. But this weak point is overcome as the resolution increases. Moreover, this process which comprises a large portion of the whole process can be improved. In this test, we didn't adapt the k-d tree to the storage of the point data. But we expect that this will help to speed up that process.

Finally, one thing that we need to remark is that in the experiment for Stanford bunny, a large amount of memory was needed for the number of point. The reason is the sparse distribution of point data. As we referred in the end of section 2, the sparsity of point data bring about a broad initial band, that is, many grid points which need to be processed. This experimental result for Stanford bunny reflects the fact.

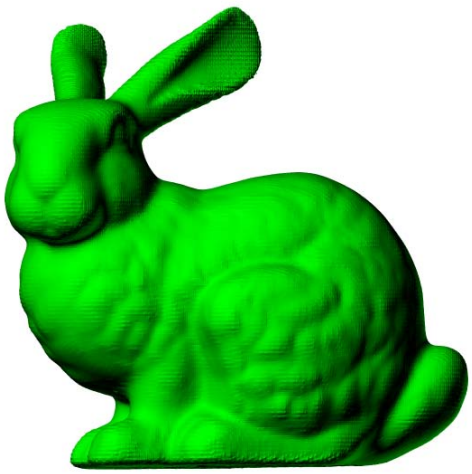

FIGURE 8. Three dimensional surface reconstruction for Stanford bunny on the grid of which maximum resolution is 256^3. The number of point is 35947.

5. CONCLUSION AND FUTURE WORK

We introduced a new surface reconstruction algorithm using level set method and octree. Although our mathematical model is based on Ye's[20], ours resolves the drawback that Ye's has. Furthermore, our algorithm is more efficient than the previous and can generate the high resolution.

Future research will address a new mathematical model which employs not Laplacian term but the term related to curvature for smoothing.

ACKNOWLEDGMENTS

Changsoo Park was supported by the Industrial Strategic Technology Program of the Ministry of Knowledge Economy(project ID:10035474). Chohong Min was supported by Priority Research Centers Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0028298). Myungjoo Kang was supported by the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2011-0002404) and Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2011(211A5020021098501007).

REFERENCES

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva, *Point Set Surfaces*, Proceedings of the conference on Visualization '01, (2001), 21–28.

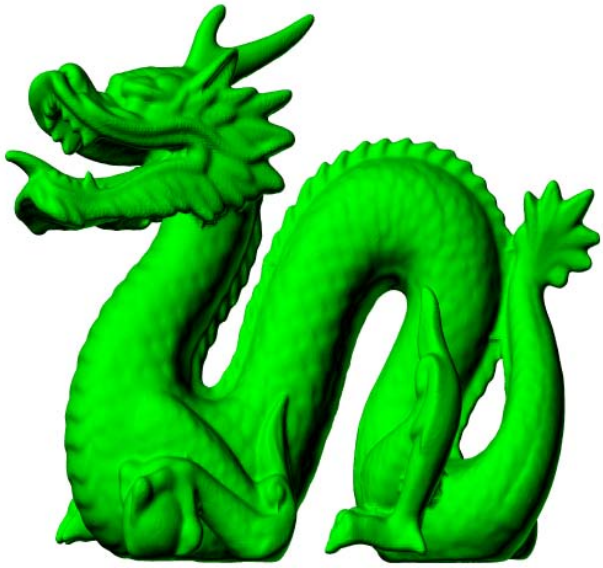

FIGURE 9. Three dimensional surface reconstruction for dragon on the grid of which maximum resolution is 256^3. The number of point is 460986.

[2] N. Amenta, M. Bern and D. Eppstein, *The crust and the β-skeleton: combinatorial curve reconstruction*, Graphical Models Image Process, **60** (1998), 125–135.

[3] N. Amenta, S. Choi, and R. Kolluri, *The power crust*, In Proc. ACM Solid Modeling, (2001), 249–260.

[4] J. C. Carr, R. K. Beatson, J. B. Cherrie and T. J. Mitchell, W. R. Fright, B. C. McCallum, T. R. Evans, *Reconstruction and Representation of 3D Objecets with Radial Basis Functions*, Proceedings of ACM SIGGRAPH, (2001), 67–76.

[5] H. Edelsbrunner and E.P. Mücke, *Three-dimensional alpha shapes*, In Workshop on Volume Visualization, (1992), 75–105.

[6] S. Fleishman, D. Cohen-Or, M. Alexa, C. T. Silva, *Progressive Point Set Surfaces*, ACM Transactions on Graphics, **22** (2003), 997–1011.

[7] T. Goldstein, X. Bresson and S. Osher, *Geometric Applications of the Split Bregman Method: Segmentation and Surface Reconstruction*, Journal of Scientific Computing, **45** (2010), 272–293.

[8] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuetzle, *Surface reconstruction from unorganized points*, Computer Graphics, **26** (1992), 71–78.

[9] X.-D. Liu, S. Osher and T. Chan, *Weighted essentially non-oscillatory schemes*, Journal of Computational Physics, **126** (1996), 202–212.

[10] F. Losasso, F. Gibou and R. Fedkiw, *Simulating Water and Smoke with an Octree Data Structure*, Proceedings of ACM SIGGRAPH, (2004), 457–462.

[11] C. Min, F. Gibou and H. Ceniceros, *A supra-convergent finite difference scheme for the variable coefficient Poisson equation on nongraded grids*, Journal of Computational Physics, **218** (2006), 123–140.

FIGURE 10. Three dimensional surface reconstruction for happy Buddha. The left one is acquired on the grid of which the maximum resolution is 256^3 and the right one does on the grid of 512^3. The number of point is 543652.

[12] C. Min and F. Gibou, *A second order accurate level set method on non-graded adaptive cartesian grids*, Journal of Computational Physics, **225** (2007), 300–321.

[13] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H.-P. Seidel, *Multi-level Partition of Unity Implicits*, Proceedings of ACM SIGGRAPH, (2003), 463–470.

[14] S. Osher and J. Sethian, *Fronts propagating with curvature dependent speed, algorithms based on a Hamilton-Jacobi formulation*, Journal of Computational Physics, **79** (1988), 12–49.

[15] S. Popinet, *Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries*, Journal of Computational Physics, **190** (2003), 572–600.

[16] H. Samet, *Applications of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990.

[17] A. Sharf, D. Alcantara, T. Lewiner, C. Greif, A. Sheffer, N. Amenta, D. Cohen-Or, *Space-time Surface Reconstruction Using Incompressible Flow*, ACM Transactions on Graphics, **27** (2008).

[18] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, Journal of Computational Physics, **77** (1988), 439–471.

[19] J. Strain, *Fast tree-based redistancing for level set computations*, Journal of Computational Physics, **152** (1999), 664–686.

[20] J. Ye, I. Yanovsky, B. Dong, R. Gandlin, A. Brandt and S. Osher, *Multigrid Narrow Band Surface Reconstruction via Level Set Functions*, CAM-Report 09-98 (2009).

[21] H. Zhao, S. Osher, B. Merriman, and M. Kang, *Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method*, Computer Vision and Image Understanding, **80** (2000), 295–314.

TABLE 2. The processing time and the required memory for reconstructing the three-dimensional Stanford bunny and dragon. (a) the time for generating octree. (b) the time for calculating the unsigned distances. (c) the time for initializing the boundary condition. (d) the time for minimizing the functional (2.8).

Data	Resolution	Uniform Grid				
		Time(sec)				Memory(MB)
		(b)	(c)	(d)	Total	
Bunny	128^3	1.03	1.72	0.57	3.32	557MB
	256^3	16.15	-	-	-	$>$ 4GB
	512^3	207	-	-	-	$\gg$ 4GB
Dragon	128^3	1.02	1.83	2.67	5.52	575MB
	256^3	16.23	-	-	-	$>$ 4GB
	512^3	205	-	-	-	$\gg$ 4GB
Happy Buddha	128^3	1.02	1.86	4.05	6.93	598MB
	256^3	15.37	-	-	-	$>$ 4GB
	512^3	204	-	-	-	$\gg$ 4GB

Data	Resolution	Octree					
		Time(sec)					Memory(MB)
		(a)	(b)	(c)	(d)	Total	
Bunny	128^3	1.69	3.54	0.12	2.14	7.49	95MB
	256^3	2.76	4.42	0.14	17.61	24.93	446MB
	512^3	5.42	5.06	0.19	144.28	154.95	1479MB
Dragon	128^3	19.24	44.89	0.18	1.16	65.47	54MB
	256^3	24.31	56.54	0.65	4.89	86.39	227MB
	512^3	31.25	66.44	0.78	45.83	144.32	1301MB
Happy Buddha	128^3	21.63	52.22	0.16	1.44	75.45	41MB
	256^3	26.67	65.25	0.54	5.55	99.01	171MB
	512^3	34.35	75.89	0.79	42.29	153.32	996MB

[22] H. Zhao, S. Osher and R. Fedkiw, *Fast Surface Reconstruction Using the Level Set Method*, Proceedings of the IEEE Workshop on Variational and Level Set Methods(VLSM'01), Washington, DC, USA 2001.

[23] H. Zhao *A FAST SWEEPING METHOD FOR EIKONAL EQUATIONS*, Mathematics of Computation, **74** (2004), 603–627.