이학박사 학위논문

Implicit Surface Reconstruction from Scattered Point Data on Octree and Feature Detection on the Implicit Surface

(팔진트리상에서 산재한 점군으로부터의 음함수 곡면 재구성과 음함수 곡면의 특징 탐지)

2013년 2월

서울대학교 대학원

수리과학부

박 창 수

Implicit Surface Reconstruction from Scattered Point Data on Octree and Feature Detection on the Implicit Surface

(팔진트리상에서 산재한 점군으로부터의 음함수 곡면 재구성과 음함수 곡면의 특징 탐지)

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2012년 10월

서울대학교 대학원

수리과학부

박 창 수

박 창 수의 이학박사 학위논문을 인준함

2012년 12월

위 원 장 _____ (인)

부 위 원 장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Implicit Surface Reconstruction from Scattered Point Data on Octree and Feature Detection on the Implicit Surface

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Changsoo Park

Dissertation Director : Professor Myungjoo Kang

Department of Mathematical Sciences
Seoul National University

February 2013

Abstract

In this thesis, we are concerned with reverse engineering process using implicit surface represented by level set. We consider two methods. One is to reconstruct implicit surface from scattered point data on octree[33] and the other detects features such as edges and corners on the implicit surface[25].

Our surface reconstruction method is based on the level set method[30] using octree i.e. a kind of adaptive grid. We start with the surface reconstruction model proposed in [46] where they considered the surface reconstruction process as an elliptic problem while most previous methods[49, 50] employed the time marching process from an initial surface to point cloud. However, as far as their method is implemented on uniform grid, it exposes inefficiency such as the high cost of memory. We improved it by adapting octree data structure[24, 23] to our problem and by introducing a new redistancing algorithm which is different from the existing one[41].

We also address feature detection from 3D CT image which is a form of implicit surface. While laser scanner is accurate and has little noise, it can't examine the inside of object. So, CT scanner is recently becoming popular for non-destructive inspection of mechanical part. But for reverse engineering, we should transform 3D image data into B-spline surface data in order to use it on CAD software, that is, change from implicit surface to parametric surface. In that process, we need feature detection for parametrization of surface. But it has more artifacts such as noise and blur than laser scanner. Consequently, preprocess for reducing artifacts is required. We apply some existing denoising algorithms to CT image data and then extract edges and corners with our feature detection method.

Key words: level set method, surface reconstruction, implicit surface, partial differential equation, octree, feature detection, reverse engineering
Student Number: 2007-30763

Contents

List of Figures

Chapter 1

Introduction

Suppose you want to design a car or mechanical parts and there is a old model that you want to model yours after and then modify it. But if it is too old for you to have any data about it to be available on computer, you cannot help starting the job at the beginning and it will take at least few weeks. However, if you have a laser scanner and reverse engineering software, the above process become far easier and you can make it in a week.

Reverse engineering is defined in [34] as follows: while forward engineering is the traditional process of moving from high-level abstractions and logical designs to the physical implementation of a system, reverse engineering is the process of duplicating an existing part, subassembly, or product, without documentation or a computer model. In the above example, reverse engineering means the process of reconstructing the complete surface from only point data, which was obtained from the laser scanner(Figure 1.1). Besides this machine design, reverse engineering is prevalent in various fields such as medicine, architecture, archeology and entertainment.

In this thesis, we pay attention to surface reconstruction, a sort of reverse engineering, from sources such as point cloud and 3D image. In chapter 2, we present surface reconstruction method from unorganized point data using octree[33]. And in chapter 3, we extract surface and detect features such as corners and edges from 3D CT volume data and they are used for reconstructing parametric surface.

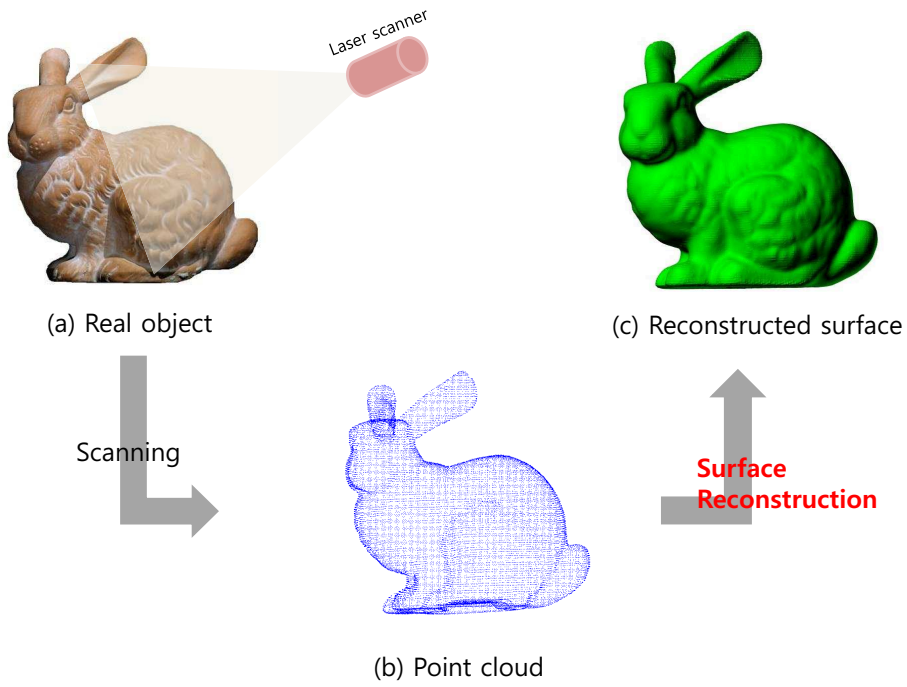


Figure 1.1: Overview of surface reconstruction from point cloud. (a) A real object is scanned with 3D laser scanner. (b) Point cloud obtained from scanning. (c) Surface reconstructed from point cloud. In the process from (b) to (c), many methods can be applied.

Chapter 2

Surface Reconstruction Method from Scattered Point Data on Octree

In this chapter, we consider a efficient method which reconstructs high resolution surface from unorganized point data[33]. This uses the level set method[30] on adaptive octree. We start from the surface reconstruction model proposed in [46]. In [46], they introduced a fast and efficient method which is different from the previous methods with the level set method. Most existing methods[49, 50] employed the time evolving process from an initial surface to point cloud. But in [46], 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 a large amount of memory because 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 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 of high resolution because the memory requirement increases geometrically.

We resolve it by adapting octree data structure[24, 23] to our problem and by introducing a new redistancing algorithm which is different from the existing one[41].

2.1 Previous work

2.1.1 History

Surface in 3D space can be reconstructed from various sources such as point cloud, 2D curves, a stack of 2D image and a 3D volume image. Among those, determining the underlying surface from point cloud is the most difficult problem because not only it is ill-posed problem but also the least information is given. Because it is so hard to find the correct solution, there were many attempts to approximate the underlying surface.

Although there are so many methodologies to do this, they are broadly classified into two categories: explicit representation and implicit representation. The former consists of Delaunay triangulations, Voronoi diagrams, the power crust[3] and so on. The latter includes Hoppe's methods[14], moving least squares[1, 11], the radial basis function[6], the level set method[49, 50], etc.

The explicit method is the method which finds a graph connecting every pair of point data. The α-shape[10], 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 the subsets of the Delaunay triangulation.

The implicit representation uses a scalar function which describes a surface as its zero level set. It is also divided into the one[14, 1, 11] which approximates the surface in the local neighborhood and the other[6, 49, 50, 46] which approximates surface globally.

The first and most famous algorithm using implicit representation is by Hoppe et al.[14]. It firstly approximates the tangent plane on a point with least squares on k nearest neighboring points. Then it finds the signed distance function on the whole domain by using the signed distance from the point to its projection onto the tangent plane.

CHAPTER 2. SURFACE RECONSTRUCTION METHOD FROM SCATTERED POINT DATA ON OCTREE

Radial basis function is very well-known function in many fields. In [6], the surface reconstruction based on radial basis function was introduced. While it can get the smooth surface, it has some limitations. It solves the linear system $Ax = b$ where the matrix A is dense and ill-conditioned. Although they employed FMM(Fast Multipole Method) for evaluation 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 was attempted by Zhao et al. in [49] and [50]. Their models make use of the evolving closed surface of which inside region includes all point data. This idea is related with the fact that the level set method came originally from computational fluid dynamics. The level set method has the advantages of handling topological changes easily and requiring no information about normal vector on point. But the previous models have the bound of speed caused by the CFL condition because they solve time-dependent PDEs such as nonlinear parabolic PDE and linear advection equation numerically.

In [46], Ye et al. approached our problem from a different standpoint. They considered the surface reconstruction as a Poisson problem. Because this made the numerical scheme free from the CFL condition, the efficiency improved compared with the previous approach. 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. It means that if numerical scheme is fulfilled only on the uniform grid, it will need so much memories. Therefore, it is almost impossible to generate the surface on the high resolution because the memory requirement increases geometrically.

The most important part in our problem is the surface passing through the neighborhood of 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 the above problem with refining only near the interface.

The adaptive grid is very efficient and prevalent in many fields such as computer graphics, scientific computing and computational fluid dynamics where they need to reduce much computational cost. Also for our problem, there are some researches[28, 37] using the adaptive grid in order to generate

high resolution. Octree is a representative adaptive grid, where the Poisson's equation can be effectively solved. The following are several Poisson solvers on octree developed up to date.

In [31], Popinet proposed a second order non-symmetric numerical method solving 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 requiring large support.

In [20], Losasso et al. proposed a first order Poisson solver using octree data structures and applied it to the Navier-Stokes equation. They stored not the pressure at nodes but 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. Consequently, most approximations in their scheme are second order accurate but approximations to the gradient of the pressure is first order accurate.

In [23], 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 gradient of the solution. The scheme employs sampling the solution at the nodes of a cell. The discretization at one cell's node uses nodes of only two or three adjacent cells, which is straightforward to implement.

Adaptive level set method was successfully developed in Min's paper [23]. Ghost nodes were treated with linear interpolation. Without increasing the support of numerical methods, they showed a possible quadratic interpolation. We apply it to our problem.

2.1.2 Fast sweeping method

When we employ the level set method, we always need the signed distance function to interface, that is, the value on each grid is required to be positive distance outside and negative distance inside from interface, respectively.

The fast sweeping method[47] is very efficient method to obtain the signed distance function. Originally, it computes the numerical solution of

the Eikonal equation

$$|\nabla u(\mathbf{x})| = f(\mathbf{x}),\ \mathbf{x} \in \mathbf{R}^n, \tag{2.1.1}$$

with boundary condition, $u(\mathbf{x}) = \phi(\mathbf{x}),\ \mathbf{x} \in \Gamma \subset \mathbf{R}^n$ on a rectangular grid. Since a specific case of Equation (2.1.1),

$$|\nabla d(\mathbf{x})| = 1,\ d(\mathbf{x}) = 0,\ \mathbf{x} \in \Gamma,$$

has the signed distance function to Γ as a solution, we can make it.

The process of the fast sweeping algorithm is as follows. Since we need just three dimensional case, the process is considered in 3D. First, Equation (2.1.1) is discretized as

$$\begin{gathered}[(u^h_{i,j,k} - u^h_{x\min})^+]^2 + [(u^h_{i,j,k} - u^h_{y\min})^+]^2 + [(u^h_{i,j,k} - u^h_{z\min})^+]^2 = f^2_{i,j,k}h^2 \\ i = 2, \ldots, I-1,\ j = 2, \ldots, J-1,\ k = 2, \ldots, K-1,\end{gathered} \tag{2.1.2}$$

where $u^h_{x\min} = \min(u^h_{i-1,j,k}, u^h_{i+1,j,k})$, $u^h_{y\min} = \min(u^h_{i,j-1,k}, u^h_{i,j+1,k})$, $u^h_{z\min} = \min(u^h_{i,j,k-1}, u^h_{i,j,k+1})$ and

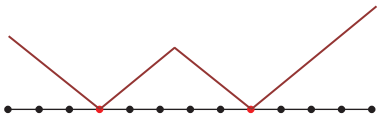
$$(x)^+ = \begin{cases} x, & x > 0, \\ 0, & x \le 0. \end{cases}$$

Next, set exact values or interpolated values at grid points in or near Γ. The values are invariant in later process. Then assign large enough positive values at other grid points. Finally, at each grid $\mathbf{x}_{i,j,k}$ whose value is not fixed in the previous step, calculate the unique solution $\bar{u}$ of Equation (2.1.2) from the current values of its neighborhoods $u^h_{i\pm1,j,k}, u^h_{i,j\pm1,k}, u^h_{i,j,k\pm1}$ and update $u^h_{i,j,k}$ with $\min(u^{\text{old}}_{i,j,k}, \bar{u})$. Zhao in [47] sweeps the whole domain with eight alternating orderings repeatedly, i.e. (1)from $i = 1$ to I, from $j = 1$ to J, from $k = 1$ to K, (2)from $i = I$ to 1, from $j = 1$ to J, from $k = 1$ to K, (3)from $i = 1$ to I, from $j = J$ to 1, from $k = 1$ to K, (4)from $i = 1$ to I, from $j = 1$ to J, from $k = K$ to 1, (5)from $i = I$ to 1, from $j = J$ to 1, from $k = 1$ to K, (6)from $i = I$ to 1, from $j = 1$ to J, from $k = K$ to 1, (7)from $i = 1$ to I, from $j = J$ to 1, from $k = K$ to 1, (8)from $i = I$ to 1, from $j = J$ to 1, from $k = K$ to 1.

The unique solution of Equation (2.1.2) is obtained as follows. First, order $u^h_{x\min}, u^h_{y\min}$ and $u^h_{z\min}$ in increasing order and let three values be a_1, a_2 and

(a) the distance function after initialization

(b) the computed distance function after the first left to right sweeping



(c) the computed distance function after the second right to left sweeping

Figure 2.1: The fast sweeping algorithm in one dimension

a_3 which satisfy $a_1 \leq a_2 \leq a_3$. Assume $a_4 = \infty$. Since there is an integer p, $1 \leq p \leq 3$, such that $\bar{x}$ is the unique solution that satisfies

$$(x - a_1)^2 + \cdots + (x - a_p)^2 = f_{i,j,k}^2 h^2 \text{ and } a_p < \bar{x} \leq a_{p+1},$$

such $\bar{x}$ and p can be obtained in the following recursive way. First, for $p = 1$, if $\tilde{x} = a_1 + f_{i,j,k}h \leq a_2$, then $\bar{x} = \tilde{x}$. Otherwise find the unique solution $\tilde{x} > a_2$ of roots for the equation

$$(x - a_1)^2 + (x - a_p)^2 = f_{i,j,k}^2 h^2.$$

If $\tilde{x}_3 \leq a_3$, then $\bar{x} = \tilde{x}$. Otherwise set the unique solution $\tilde{x} > a_3$ of roots for the equation

$$(x - a_1)^2 + (x - a_p)^2 + (x - a_3)^2 = f_{i,j,k}^2 h^2$$

to $\bar{x}$.

The fast sweeping method requires the computational time of $O(N)$, where N is the size of grid on the whole domain. As far as we know, it is the fastest algorithm for calculating signed distances.

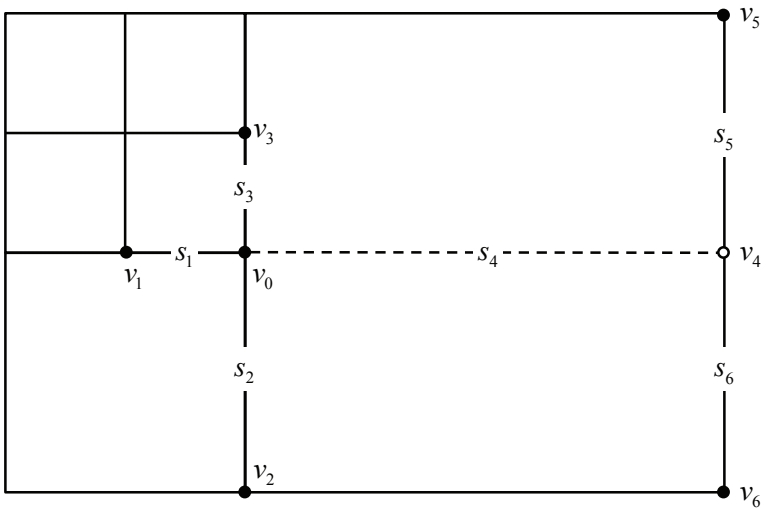


Figure 2.2: Neighboring nodes of a T-junction node v_0 in two dimensions. v_4 means the ghost neighboring node of v_0.

2.1.3 Basic finite difference methods on octree

From now on, we denote a (rectangular) cell in quadtree or octree by C and the node of cell by v.

Usually when we employ adaptive grid, the main difficulty comes from discretizations at T-junction nodes, i.e. nodes for which there is a missing neighboring node in the directions along one of coordinate axes. For example, Figure 2.2 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 the positive x-direction. Here, the value of a node-sampled function $\phi_i : \{\mathbf{v}_i\} \to \mathbb{R}$ at the ghost node $\mathbf{v}_4$ can be defined by the following linear interpolation:

$$\phi_4^G = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6}. \tag{2.1.3}$$

However, instead of this second order accurate interpolation, Min et al. in [24] use the following third order accurate interpolation. First, a simple Taylor expansion demonstrates that the interpolation error in Equation (2.1.3) is given by

$$\phi_4^G = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6} = \phi(\mathbf{v}_4) + \frac{s_5 s_6}{2}\phi_{yy}(\mathbf{v}_0) + O(\triangle x_{\text{smallest}})^3, \tag{2.1.4}$$

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 by 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 (2.1.4) to give

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

In the above interpolation, employing the node values of only the cells adjacent to $\mathbf{v}_0$ helps the efficient performance since referring cells not immediately adjacent to the current cell requires more difficult process.

In three spatial dimensions, they use similar interpolation procedures in order to define the value of ϕ at ghost nodes. Referring to Figure 2.3, a T-junction node $\mathbf{v}_0$ has four regular neighboring nodes and two ghost nodes. The values of a node-sampled function $\phi_i : \{\mathbf{v}_i\} \to \mathbb{R}$ at the ghost nodes $\mathbf{v}_4$ and $\mathbf{v}_5$ can be defined by second order linear and bilinear interpolations as follows:

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

In the same way to quadtree, third order accurate interpolations can be derived by canceling out the second order derivatives in the error term to

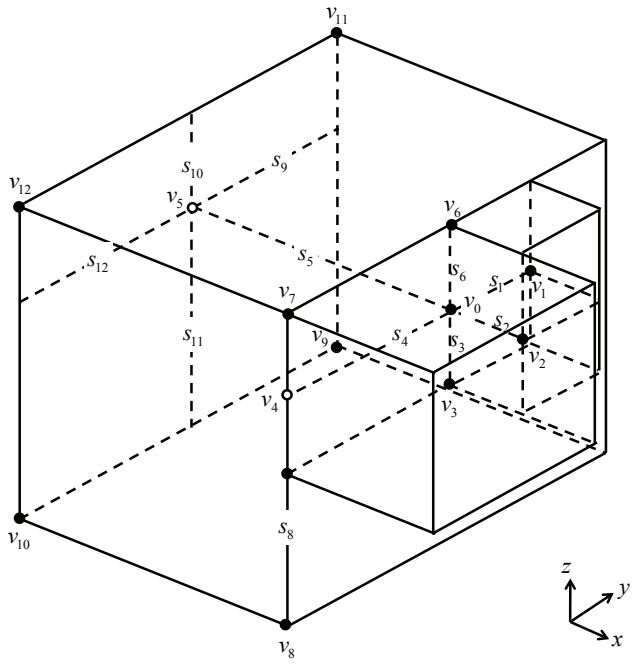


Figure 2.3: Neighboring vertices of a vertex v_0 in three spatial dimensions. v_4 and v_5 mean the ghost neighboring nodes of v_0.

obtain

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

The third order interpolations defined above facilitate treating T-junction nodes in a same fashion as a regular node, up to third order accuracy, where a regular node means a node for which all the neighboring nodes in the direction along coordinate axes exist. Therefore, finite differences for ϕ_x, ϕ_y, ϕ_z, ϕ_{xx}, ϕ_{yy} and ϕ_{zz} at every nodes with standard finite difference formulas can be defined in a dimension by dimension framework. For instance, referring to Figure 2.4, the central difference formulas for ϕ_x and ϕ_{xx} are as follows:

$$D_x^0\phi_0 = \frac{\phi_2 - \phi_0}{s_2}\cdot\frac{s_1}{s_1 + s_2} + \frac{\phi_0 - \phi_1}{s_1}\cdot\frac{s_2}{s_1 + s_2},$$
$$D_{xx}^0\phi_0 = \frac{\phi_2 - \phi_0}{s_2}\cdot\frac{2}{s_1 + s_2} - \frac{\phi_0 - \phi_1}{s_1}\cdot\frac{2}{s_1 + s_2}.$$

Also, the forward and backward first order accurate approximations for the first order derivatives are

$$D_x^+\phi_0 = \frac{\phi_2 - \phi_0}{s_2},$$
$$D_x^-\phi_0 = \frac{\phi_0 - \phi_1}{s_1}$$

and the second order accurate approximations for the first order derivatives are

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

Figure 2.4: One dimensional adaptive grid.

where they use the minmod slope limiter [19, 40] because it produces more stable results in region where ϕ might present kinks. Approximations in the y and z directions can be also obtained similarly.

2.1.4 Biconjugate gradient stabilized(BICGSTAB) algorithm

Consider the linear system

$$A\mathbf{x} = \mathbf{b}, \tag{2.1.5}$$

where A is $n \times n$.

A general projection method for solving (2.1.5) extracts an approximate $\mathbf{x}_m$ from an affine subspace $\mathbf{x}_o + \mathcal{K}_m$ of dimension m by imposing the Petrov-Galerkin condition

$$\mathbf{b} - A\mathbf{x}_m \perp \mathcal{L}_m,$$

where $\mathcal{L}_m$ is another subspace of dimension m.

When A is symmetric positive definite, the Conjugate Gradient(CG) algorithm is one of the best projection method. But if A is nonsymmetric, we cannot exploit CG.

The Biconjugate Gradient(BCG) algorithm is a sort of Krylov subspace method for which the subspace $\mathcal{K}_m$ is the Krylov subspace

$$\mathcal{K}_m(A, \mathbf{v}_1) = \text{span}\{\mathbf{v}_1, A\mathbf{v}_1, A^2\mathbf{v}_1, \ldots, A^{m-1}\mathbf{v}_1\}$$

and the subspace $\mathcal{L}_m$ is the Krylov subspace

$$\mathcal{K}_m(A^T, \mathbf{w}_1) = \text{span}\{\mathbf{w}_1, A^T\mathbf{w}_1, (A^T)^2\mathbf{w}_1, \ldots, (A^T)^{m-1}\mathbf{w}_1\},$$

where $\mathbf{v}_1 = \mathbf{r}_0 / \parallel \mathbf{r}_0 \parallel_2$, $\mathbf{r}_0 = b - A\mathbf{x}_0$ and $\mathbf{w}_1$ is arbitrary but $(\mathbf{v}_1, \mathbf{w}_1) \neq 0$.

Although this algorithm is useful for nonsymmetric A, it has the drawback that A^T is required. Because there are many situations where A^T is not available, the use of A^T should be bypassed.

The Conjugate Gradient Squared(CGS) algorithm[36] was developed in order to avoid using the A^T in BCG. They in [36] removed the calculation of A^T by squaring residual polynomials and adding some recurrences. However, the CGS algorithm also has the limitation that it gives rise to substantial rounding errors in the case of irregular convergence since it is based on squaring the residual polynomials.

The Biconjugate Gradient Stabilized algorithm[43] remedied this problem by adopting a new polynomial defined recursively at each step in order to stabilize or smooth the convergence behavior of the original CGS algorithm. The detail of algorithm is as follows.

Algorithm 1 BICGSTAB

1. Compute $\mathbf{r}_0 := \mathbf{b} - A\mathbf{x}_0,\ \mathbf{r}_0^*$ arbitrary
2. $\mathbf{p}_o := \mathbf{r}_0$
3. For $j = 0, 1, \ldots,$ until convergence, Do
4. $\quad \alpha_j := (\mathbf{r}_j, \mathbf{r}_0^*)/(A\mathbf{p}_j, \mathbf{r}_0^*)$
5. $\quad \mathbf{s}_j := \mathbf{r}_j - \alpha_j A\mathbf{p}_j$
6. $\quad w_j := (A\mathbf{s}_j, \mathbf{s}_j)/(A\mathbf{s}_j, A\mathbf{s}_j)$
7. $\quad \mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j + w_j \mathbf{s}_j$
8. $\quad \mathbf{r}_{j+1} := \mathbf{s}_j - w_j A\mathbf{s}_j$
9. $\quad \beta_j := \frac{(\mathbf{r}_{j+1}, \mathbf{r}_0^*)}{(\mathbf{r}_j, \mathbf{r}_0^*)} \times \frac{\alpha_j}{w_j}$
10. $\quad \mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j(\mathbf{p}_j - w_j A\mathbf{p}_j)$
11. End Do

2.2 Mathematical models

The most well-known variational model for reconstructing surface with the level set method is the weighted minimal surface model[49]. Let $\mathcal{S}$ denote the data set which can include point data, pieces of curves or surfaces. In our problem, $\mathcal{S}$ means point data. 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 starts with the following surface energy:

$$E(\Gamma) = \left[\int_\Gamma d^p(\mathbf{x}) ds \right]^{\frac{1}{p}}, \quad 1 \leq p \leq \infty, \tag{2.2.1}$$

where Γ is an arbitrary surface and ds is the surface area. To find the surface which minimizes the surface energy (2.2.1), they calculate the gradient descent of the functional (2.2.1).

$$\frac{d\Gamma}{dt} = -\left[\int_\Gamma d^p(\mathbf{x}) ds \right]^{\frac{1}{p}-1} d^{p-1}(\mathbf{x}) \left[\nabla d(\mathbf{x}) \cdot \mathbf{n} + \frac{1}{p} d(\mathbf{x}) \kappa \right] \mathbf{n}, \tag{2.2.2}$$

where $\mathbf{n}$ is the unit outward normal and κ is the mean curvature.

The motion of surface from the above gradient flow (2.2.2) is described as follows. Initially, a closed surface including all point data inside should be given. Then the surface shrinks toward point set with keeping the balance between the potential force $\nabla d(x) \cdot \mathbf{n}$ and the surface tension $d(\mathbf{x})\kappa$. The term $\nabla d(x) \cdot \mathbf{n}$ forces the surface to move in the direction of points and the term $d(\mathbf{x})\kappa$ influences the smoothness of the surface. When all point data lies on the surface, the surface energy is minimized and the surface stops evolving. The scalar function $d(\mathbf{x})$ makes the surface more flexible in regions close to the data set and more rigid in regions distant from the data set.

Here, the level set method[30] is used for obtaining an implicit surface and handling topological changes. The level set method is very popular in computational fluid dynamics, computer graphics and image processing because of its advantage of handling the complicated topology and implementing easily. It represents the contour or the surface as the zero level set of a higher dimensional signed distance function. We explain the detail as follows.

Let $\Omega(t)$ be the (generally multiply connected) region enclosed by $\Gamma(t)$. Then the level set method uses the level set function $\phi(\vec{x}, t)$ to represent $\Gamma(t)$ as the zero level set of $\phi(\vec{x}, t)$ as shown in Figure 2.5, i.e.

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

Then, as [49], level set formulation of (2.2.1) is

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

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 (2.2.2) is

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| \left[\int d^p(\mathbf{x}) \delta(\phi) |\nabla \phi| d\mathbf{x} \right]^{\frac{1}{p}-1} d^{p-1}(\mathbf{x}) \left[\nabla d(\mathbf{x}) \cdot \frac{\nabla \phi}{|\nabla \phi|} + \frac{1}{p} d(\mathbf{x}) \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right], \tag{2.2.3}$$

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

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

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

$$\frac{d\Gamma(t)}{dt} = -\nabla d(\mathbf{x}). \tag{2.2.4}$$

The level set formulation of Equation (2.2.4) is

$$\frac{\partial \phi}{\partial t} = \nabla d(\mathbf{x}) \cdot \nabla \phi. \tag{2.2.5}$$

The evolution equation (2.2.3) is a nonlinear parabolic equation because it has the term for the mean curvature of the surface. Since the convection equation (2.2.5) 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 restriction.

The above two models handle the surface reconstruction problem as the time evolution equation. In [46], Ye et al. considered our problem as Poisson's

equation. They introduced the following energy functional for ϕ : Given point data $\{\mathbf{x}_l\}_{l=1,\dots,N} \subset \mathcal{S}$,

$$E(\phi) = \int G(\phi(\mathbf{x}))d\mathbf{x} + \sum_{l=1}^{J} \beta_l (P_l\phi)^2, \tag{2.2.6}$$

where $P_l\phi = \int p_l(\mathbf{x})\phi(\mathbf{x})d\mathbf{x}$ is the projection operator 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 in (2.2.6) plays a role as the diffusion term which influences the smoothness of the surface and the second term functions as the fidelity term which fits the surface as close as possible to point data. The weight parameter β_l in the second term affects the extent of fitting the surface. If a large enough β_l is chosen, even crude initial boundary conditions result in very successful fitting.

Instead of getting the Euler-Lagrange equation from the energy functional (2.2.6) on continuous domain directly, they considered the discretized version of Equation (2.2.6). In the case of two-dimensional space, the equation is as follows.

$$\bar{E}(\phi) = \sum_i \sum_j \left(\frac{\phi_{i+1,j} - \phi_{i,j}}{h}\right)^2 + \left(\frac{\phi_{i,j+1} - \phi_{i,j}}{h}\right)^2 + \sum_l \beta_l [\bar{P}_l\phi]^2, \tag{2.2.7}$$

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 (2.2.7) with respect to $\phi(i,j)$ at the grid point (i,j), they obtained the Euler-Lagrange equation as follows

$$\frac{1}{2}\frac{\delta\bar{E}}{\delta\phi_{i,j}} = -\frac{\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j}}{h^2} + \sum_{l=1}^{J} \beta_l p^l_{i,j} P_l\phi = 0, \tag{2.2.8}$$

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 $\mathbf{x}_l$.

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

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 dimensional space, this model can be extended with 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 through solving the Poisson's equation (2.2.8).

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

However, their method still has the limitation that it needs so much memories because it can be implemented 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. Of course, 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.

Eventually, we need adaptive grid because it results 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[23].

2.3 Numerical method

2.3.1 Tree generation and splitting condition

We use the standard quadtree(resp. octree) data structure to represent the spatial discretization of the two(resp. three)-dimensional domain. The generation of tree is described as follows. For a point $\mathbf{x}_1$, it is checked whether the root cell C of tree corresponding to the whole domain satisfies the splitting condition

$$\text{dist}(\mathbf{x}_l, \partial C) \leq \frac{1}{2}\min\{\text{width of } C, \text{height of } C\}. \tag{2.3.1}$$

If so, the root cell is divided into four(resp. eight) children cells. Otherwise, the splitting stops. And then we check if each child cell C meets the splitting condition (2.3.1). If so, the cell C is split. Otherwise, the cell C is not split and doesn't have its children cells any more. This process is repeated until the desired level of detail is achieved. More specifically, splitting stops after the finest resolution for cells containing at least a point is fulfilled.

After finishing the splitting for $\mathbf{x}_1$, we repeat the above splitting process for another point $\mathbf{x}_2$. Of course, during this process, we exclude cells which are already split in the previous step.

In our problem, we use the graded grids, which limits the difference of level between two adjacent cells to at most one, in order to guarantee the uniform grid near the interface. Otherwise, incorrect information on the nodes of large cells compared with neighboring cells may corrupt values at distant nodes as well as neighboring nodes. The graded grids and the nongraded grids are demonstrated in Figure 2.7.

Lemma 2.3.1. *The splitting condition (2.3.1) 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 $\mathbf{x}$ is inside a cell, it is obvious that the condition (2.3.1) is satisfied. Thus the finest resolution is guaranteed for the cell containing the point $\mathbf{x}$. We denote the finest cell containing the point $\mathbf{x}$ by C_c. For cells not having the point $\mathbf{x}$, 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) and 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 (2.3.1). 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.

□

2.3.2 Distance function

After the generation of octree, we calculate the distance from point data $\{\mathbf{x}_l\}$ 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 child of the root cell.

$$\mathrm{dist}(\mathbf{x}_l, \partial C) \leq \max\{\sqrt{(2l_1)^2 + l_2^2}, \sqrt{l_1^2 + (2l_2)^2}\} + \min\{l_1, l_2\}, \qquad (2.3.2)$$

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 (2.3.2) 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 for another point $\mathbf{x}_2$, we update the distance at each node of the cells satisfying the above condition (2.3.2). It stops after looking all points $\{\mathbf{x}_l\}$ over.

In the process of updating the distances, it is better to search as many nodes as possible because there may be nodes of which distances are not minimum. Of course, if we choose region large enough, the updating process will assure the minimum distance at all nodes. But it gets too time-consuming

chore. Thus we need the least area guaranteeing a complete distance function in the entire domain. The condition (2.3.2) make it.

Determining the distance on the octree is more efficient compared to that on the uniform grid. Denote the one side resolution of the uniform grid discretizing 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 [47] 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 of some region. Then the time of distancing on the uniform grid is approximately $O(LN)$ while $O(L\log(N))$ in the octree. We will demonstrate this result in Section 2.4.

2.3.3 Initial guess of signed distance function

Once the above processes are completed, we need to initialize boundary condition in order to solve the Poisson's equation. The 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 signed distance function ϕ. To achieve it, we employ the fast tagging method in [50] 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 [46], we start from a node $\mathbf{v}_1$ on the boundary of the entire domain in order to mark the outside region. First, the node $\mathbf{v}_1$ is marked as the outside. Then we check if the distances at neighboring nodes $\{\mathbf{v}_{N_i}\}_{N_i=1,..,8}$ of $\mathbf{v}_1$ satisfy the condition $d(\mathbf{v}_{N_i}) > \epsilon$ where ϵ is a constant relative to bandwidth. 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 get the outside region. Then, we set the distances in the outside region to ϵ and the distances in $\{\mathbf{x} : d(\mathbf{x}) < \epsilon\}$ to 0. Finally, the remaining nodes become the inside region, where the distances 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. But this choice may give rise to problems. For some data sets, i.e. a set having a big hole, the ϵ_{data} may be large. Then initial guess can be very rough because of large bandwidth. In this case, our method cannot guarantee appropriate result.

2.3.4 Numerical discretization of model (2.2.6) on octree

We discretize Equation (2.2.6) based on the scheme in Section 2.1.3. While the coefficients of the discretized fitting term on the octree differ little from those 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 the octree. Let us consider Figure 2.3, where only $\mathbf{v}_5$ was regarded as the ghost node which needs bilinear interpolation. We extend this situation to all neighboring nodes of $\mathbf{v}_0$, that is, $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$ and $\mathbf{v}_6$ as shown in Figure 2.10. Then like 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.

$$\begin{aligned}
\phi_1^G &= \frac{s_{18}s_{20}\phi_{17} + s_{18}s_{19}\phi_{18} + s_{17}s_{20}\phi_{19} + s_{17}s_{19}\phi_{20}}{(s_{17}+s_{18})(s_{19}+s_{20})} \\
&= \phi(\mathbf{v}_1) + \frac{1}{2}(s_{17}s_{18}\phi_{zz}(\mathbf{v}_0) + s_{19}s_{20}\phi_{xx}(\mathbf{v}_0)) \\
\phi_2^G &= \frac{s_{13}s_{16}\phi_{13} + s_{14}s_{16}\phi_{14} + s_{13}s_{15}\phi_{15} + s_{14}s_{15}\phi_{16}}{(s_{13}+s_{14})(s_{15}+s_{16})} \\
&= \phi(\mathbf{v}_2) + \frac{1}{2}(s_{15}s_{16}\phi_{zz}(\mathbf{v}_0) + s_{13}s_{14}\phi_{yy}(\mathbf{v}_0)) \\
\phi_3^G &= \frac{s_{28}s_{29}\phi_{27} + s_{27}s_{29}\phi_{28} + s_{28}s_{30}\phi_{29} + s_{27}s_{30}\phi_{30}}{(s_{27}+s_{28})(s_{29}+s_{30})} \\
&= \phi(\mathbf{v}_3) + \frac{1}{2}(s_{27}s_{28}\phi_{xx}(\mathbf{v}_0) + s_{29}s_{30}\phi_{yy}(\mathbf{v}_0)) \\
\phi_4^G &= \frac{s_8s_{22}\phi_7 + s_8s_{21}\phi_8 + s_7s_{22}\phi_{21} + s_8s_{21}\phi_{22}}{(s_7+s_8)(s_{21}+s_{22})} \\
&= \phi(\mathbf{v}_4) + \frac{1}{2}(s_7s_8\phi_{zz}(\mathbf{v}_0) + s_{21}s_{22}\phi_{xx}(\mathbf{v}_0))
\end{aligned}$$

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

Then, we obtain the following equations through simple manipulation.

$$\begin{aligned}\left(\frac{\phi_2^G-\phi_0}{s_2}+\frac{\phi_5^G-\phi_0}{s_5}\right)\frac{2}{s_2+s_5} &= \phi_{xx}(\mathbf{v}_0) + \left(\frac{s_9s_{12}}{(s_2+s_5)s_5}+\frac{s_{13}s_{14}}{(s_2+s_5)s_2}\right)\phi_{yy}(\mathbf{v}_0) \\ &+ \left(\frac{s_{10}s_{11}}{(s_2+s_5)s_5}+\frac{s_{15}s_{16}}{(s_2+s_5)s_2}\right)\phi_{zz}(\mathbf{v}_0) \qquad (2.3.3)\\ \left(\frac{\phi_4^G-\phi_0}{s_4}+\frac{\phi_1^G-\phi_0}{s_1}\right)\frac{2}{s_1+s_4} &= \left(\frac{s_{19}s_{20}}{(s_1+s_4)s_1}+\frac{s_{21}s_{22}}{(s_1+s_4)s_4}\right)\phi_{xx}(\mathbf{v}_0) + \phi_{yy}(\mathbf{v}_0) \\ &+ \left(\frac{s_{17}s_{18}}{(s_1+s_4)s_1}+\frac{s_7s_8}{(s_1+s_4)s_4}\right)\phi_{zz}(\mathbf{v}_0) \qquad (2.3.4)\\ \left(\frac{\phi_6^G-\phi_0}{s_6}+\frac{\phi_3^G-\phi_0}{s_3}\right)\frac{2}{s_3+s_6} &= \left(\frac{s_{27}s_{28}}{(s_3+s_6)s_3}+\frac{s_{25}s_{26}}{(s_3+s_6)s_6}\right)\phi_{xx}(\mathbf{v}_0) \\ &+ \left(\frac{s_{29}s_{30}}{(s_3+s_6)s_3}+\frac{s_{23}s_{24}}{(s_3+s_6)s_6}\right)\phi_{yy}(\mathbf{v}_0) + \phi_{zz}(\mathbf{v}_0) \qquad (2.3.5)\end{aligned}$$

By multiplying equations (2.3.3),(2.3.4),(2.3.5) 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.

2.4 Results

We tested our method and Ye's[46] 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 adapted the Biconjugate Gradient Stabilized (BICGSTAB) algorithm[35, 43] to the process of minimizing the functional (2.2.6). In the case of the uniform grid, we can guarantee that the matrix obtained from numerical scheme is symmetric. However, as long as we use the adaptive grid, we cannot guarantee the matrix is symmetric. That's the why we adopt the BICGSTAB algorithm. The stopping criterion used is that 2-norm of the residual is less than 10^{-8}.

2.4.1 Five-leafed clover

First, we generated a set of artificial point data of which shape is five-leafed clover on two-dimension. The data are 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. Of course, our method can be implemented with quadtree in two-dimension. We compares the processing time and the required memory on the quadtree with those on the uniform grid in Table 2.1. Like Table 2.1 shows, the processing time on the quadtree gets shorter than that on the uniform grid as the resolution increases. Furthermore, the difference of the memory requirement is much greater than the difference of the time.

In this example, the qualities of the curves reconstructed on several different resolutions made no difference because overall shape was comparatively smooth. So we demonstrate just one of those in Figure 2.12. Figure 2.11 represents the generation of quadtree and the initial narrow band before solving the minimization of the functional (2.2.6).

2.4.2 Bunny, Dragon, Happy buddha

In three dimensional, we reconstructed bunny, dragon and happy buddha from Stanford 3D Scanning Repository. Figure 2.13 and Figure 2.14 show results on grids of which maximum resolution is 256^3 for bunny and dragon, respectively. Actually, although we obtained the result of 1024^3 resolution with the 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 can confirm the improvement of detail as the resolution increases in the case of happy buddha(Figure 2.15).

In Table 2.2, 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 the octree. In case the resolution was 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 structure also saved much processing time compared with the uniform grid as the resolution get higher. Moreover, the process of calculating

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

Table 2.1: The processing time and the required memory for reconstructing the two-dimensional five-leafed clover

the unsigned distances 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 point data. But we expect that it will help improve speed in the process.

Finally, one thing that we need to remark is that in the experiment for bunny, a large amount of memory was required while the number of point data is small relatively. The reason is the sparse distribution of point data. As we referred in the end of Section 2.2, the sparsity of point data bring about a broad initial band, that is, many grid points which need to be processed. This experimental result reflects it.

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

Table 2.2: The processing time and the required memory for reconstructing the three-dimensional 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 (2.2.6).

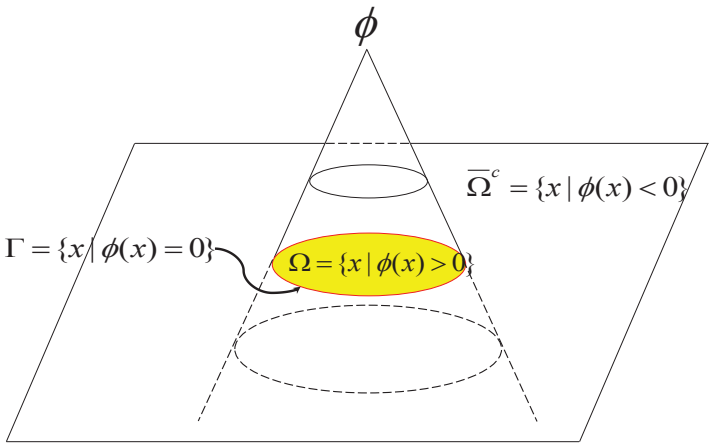


Figure 2.5: Level Set function ϕ for representing Γ on Ω.

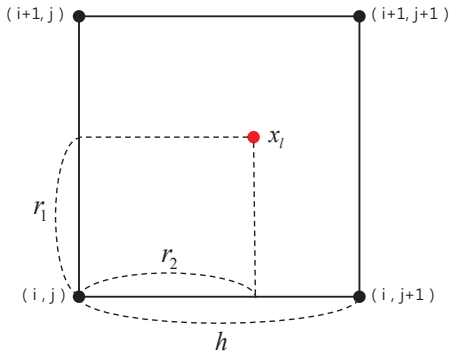


Figure 2.6: The description of 2D interpolation on the uniform grid

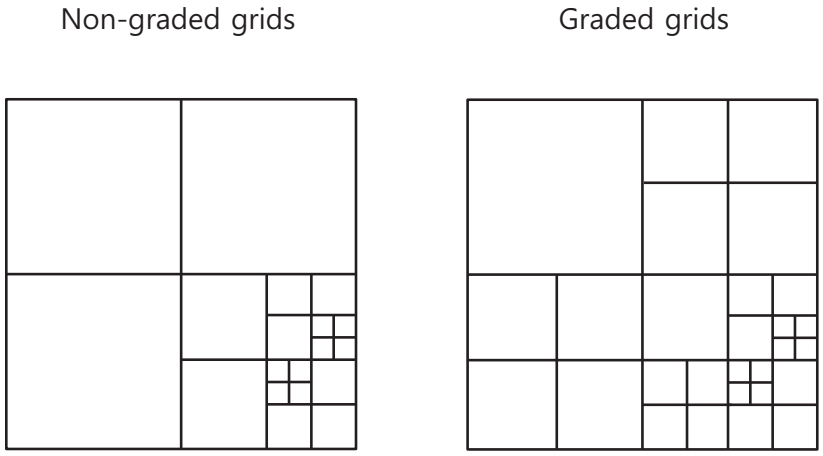


Figure 2.7: The left picture demonstrates nongraded grids and the right one demonstrates graded grids.

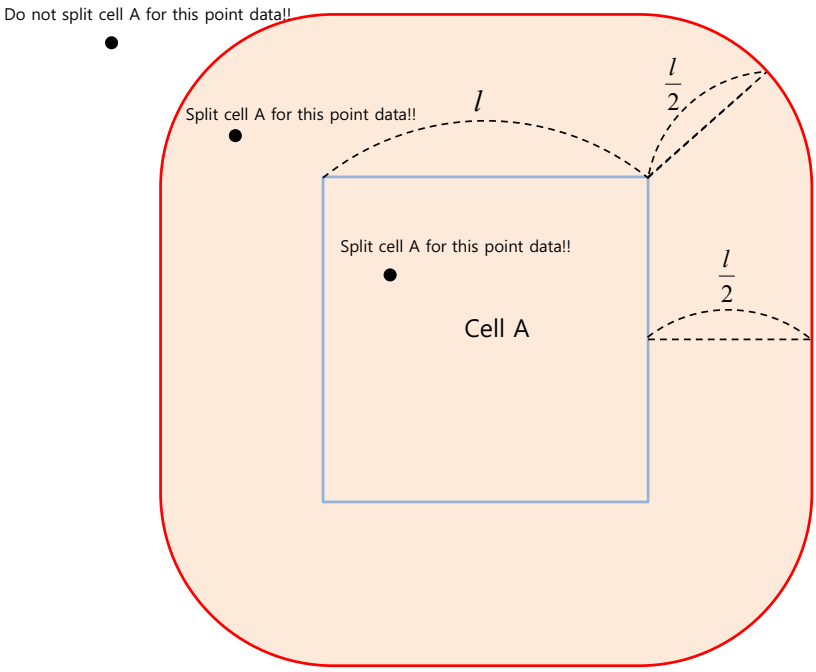


Figure 2.8: The light orange region demonstrates the location of point satisfying the splitting condition (2.3.1) for the cell A.

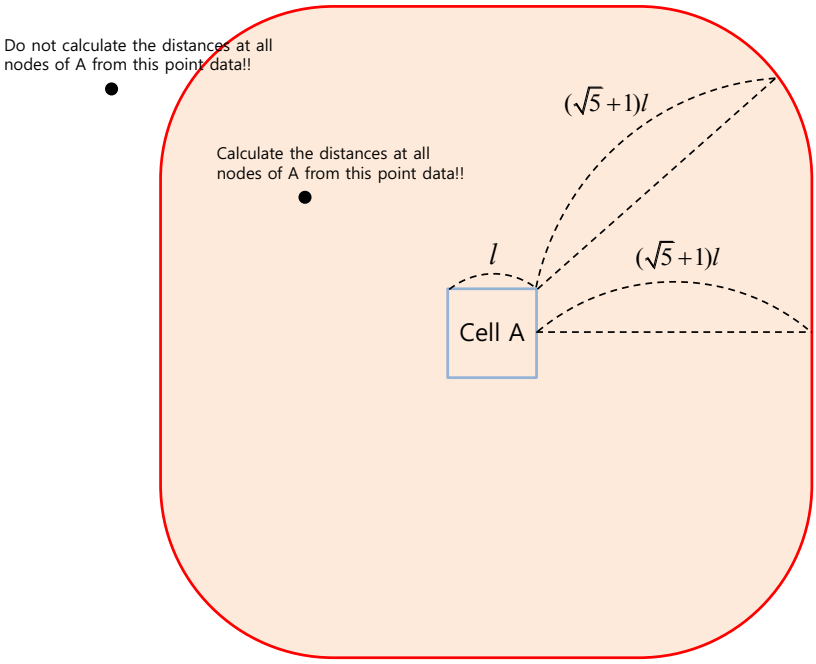


Figure 2.9: The light orange region demonstrates the location of point satisfying the distance updating condition (2.3.2) for the cell A.

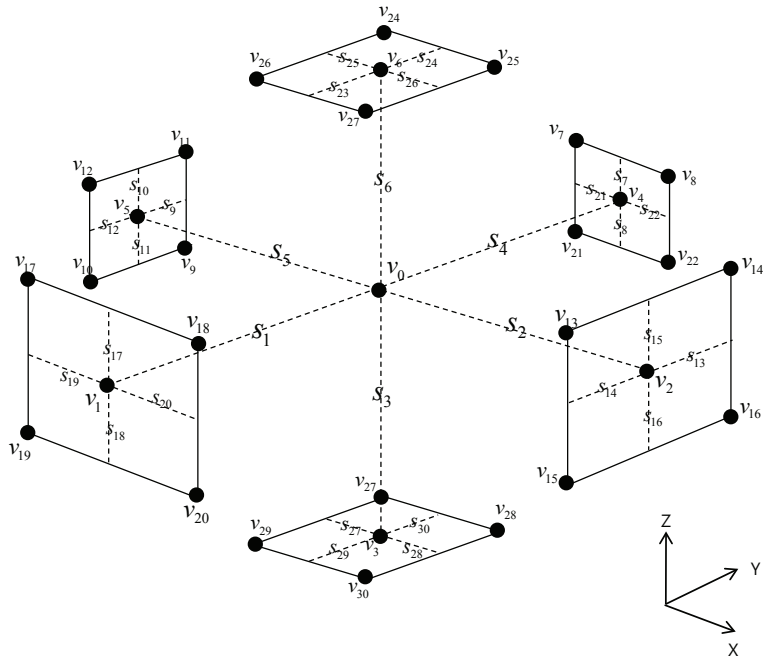


Figure 2.10: We consider all neighboring nodes of a vertex $\mathbf{v}_0$ as the ghost nodes in our scheme. But this situation never happens actually.

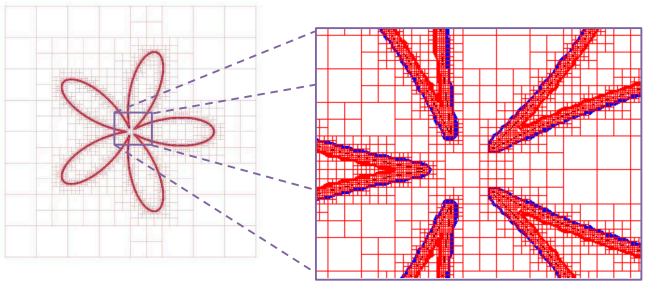


Figure 2.11: The left picture represents the initial narrow band with the boundary condition. The right picture is the magnified concave part of five-leafed clover. The red thick lines denote the outside boundaries and the blue thick line denote the inside boundaries. The black dots represent point data. 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.

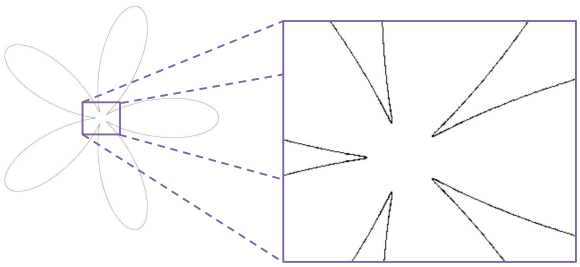
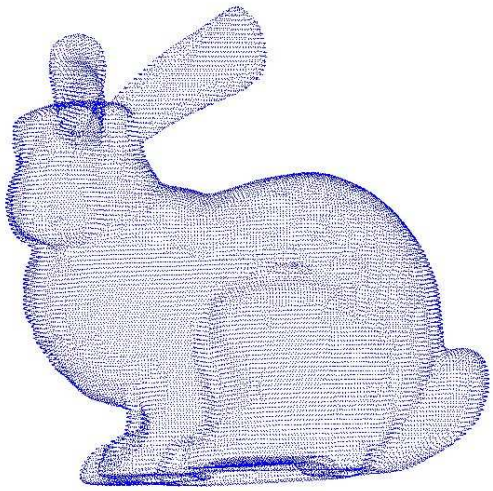
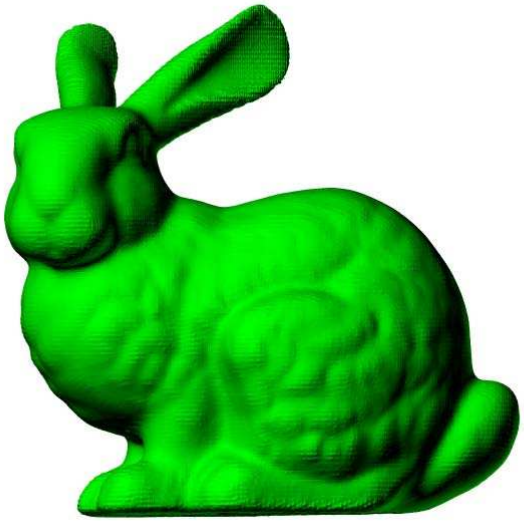


Figure 2.12: The left picture shows the reconstructed five-leafed clover from point cloud. The right picture is the magnified concave part of five-leafed clover. This indicates that the curve is well-fitted to even the structure with high curvature.

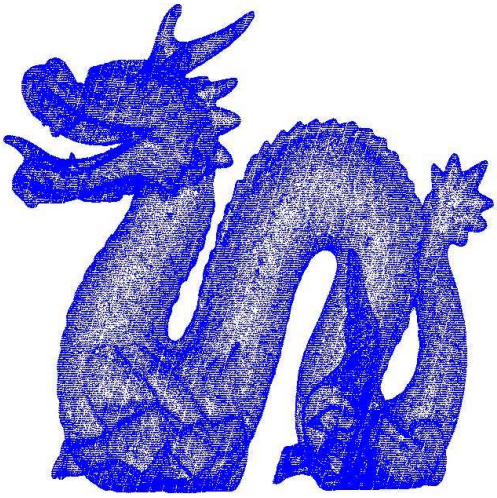


(a) point cloud



(b) reconstructed surface

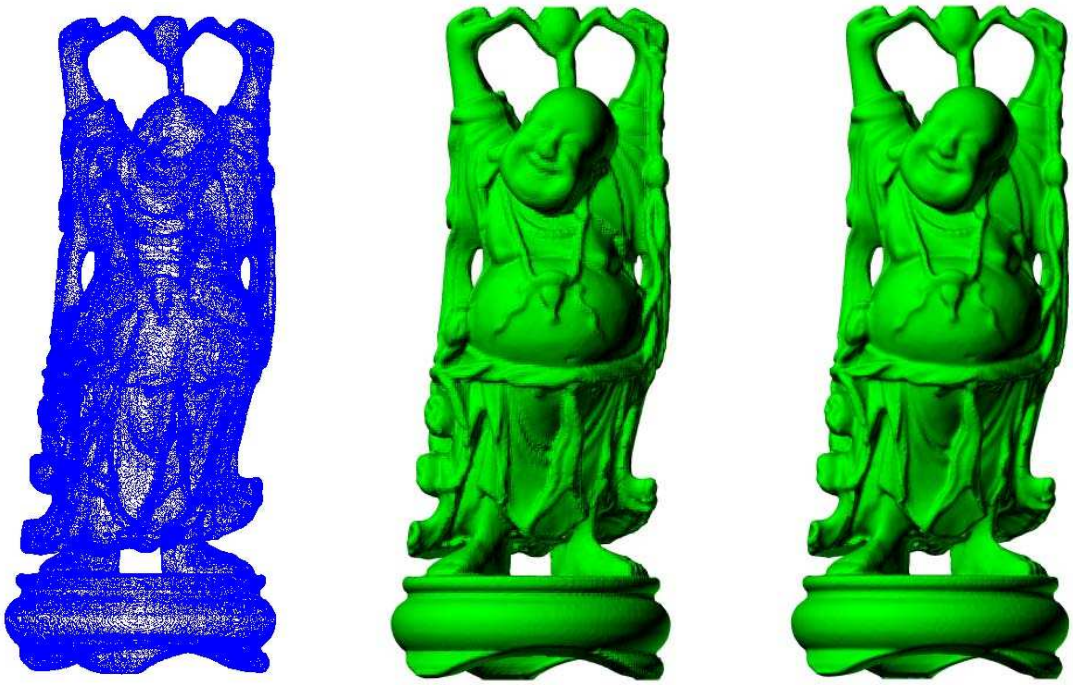
Figure 2.13: Three dimensional surface reconstruction for bunny on the grid of which maximum resolution is 256^3. The number of point is 35947.



(a) point cloud

(b) reconstructed surface

Figure 2.14: Three dimensional surface reconstruction for dragon on the grid of which maximum resolution is 256^3. The number of point is 460986.



(a) point cloud (b) reconstructed surface

Figure 2.15: Three dimensional surface reconstruction for happy buddha. The left one in (b) is acquired on the grid of which the maximum resolution is 256^3 and the right one in (b) does on the grid of 512^3. The number of point is 543652.

Chapter 3

Feature Detection on Implicit Surface

Recently, CT(Computed Tomography) scan is getting more popular in mechanical engineering because it enables nondestructive testing of mechanical parts. So it is a hot issue to transform 3D volume image data from CT scan into parametric surfaces such as B-spline model(Figure 3.1). In this process, a sort of reverse engineering, the most important part is to divide the whole domain into the set of appropriate domains for parametrization. In order to accomplish it, we need to extract the information about features such as corners, sharp edges, ridges. Feature detection has attracted lots of attention [15, 45, 44, 8, 16, 32, 22, 13, 5, 17, 4]. However, the existing methods either depend on mesh generation or take much time because of checking all voxel data. Also, extracting features from CT scan data with accuracy is very tough since CT scan data has much more artifacts than laser scan data. So it vitally requires the preprocessing task to reduce artifacts.

In this chapter, we address a efficient method detecting features from CT scan data[25]. Since our method extracts features from not 3D volume data but 2D surface data, operation time can be saved. Also with diverse preprocessing steps, it can remove artifacts considerably.

The overview of process is as follows(Figure 3.2). First, we apply the segmentation algorithm[39] using the level set method to CT scanned image in order to get 3D binary image data which has two different values inside

and outside respectively. Although we can get the 2D surface from the binary image easily, we transform this to the signed distance function with fast sweeping method[47]. Then with simple thresholding for the signed distance function, we can focus on not the whole volume but the surface in the process of detecting features. Next, in order to reduce noises, we adapt nonlocal means filtering[9] to the segmented surface. Because this filter has the advantages of preserving edges effectively as well as removing noises on the surface, it helps to detect the edge and corners more accurately. Finally, corners and sharp edges are determined. The corners are detected based on Sobel-like mask convolution with the corner candidates obtained from the idea of the marching cubes algorithm. The sharp edges are extracted based on new-patterned mask convolution using the property of the signed distance function. And the post processing using the idea of SUSAN(Smallest Univalue Segment Assimilating Nucleus) method[38] is employed for removing incorrect features.

3.1 Related work and background

Feature detection method on 3D data can be classified into two groups: polygon-based method and point-based method. There exist many techniques for feature detection relying on polygonal meshes[15, 44, 16, 22]. These techniques often consist of two steps such as mesh generation and feature detection. In [16, 22], discrete extremities are used for extracting feature lines in regular triangles. Singular triangles are treated specially based on neighbors of singular or regular triangles. In [15], a normal-based classification operator was used in order to classify edges into different types by combining some thresholds. In [44], Watanabe and Belyaev estimated the principle curvature extrema on dense triangular meshes. Then, they utilized these curvature extrema for constructing focal surface, of which the properties were used for identification of features.

Point based methods[45, 8, 32, 13] are more interesting because of lack of knowledge about the normal vector and connectivity information. Gumhold et al. in [13] considered the k-nearest neighbors for every data voxel. They used Principal Component Analysis (PCA) to analyze the neighbors of each

point. Eigenvalues and eigenvectors of coefficient matrix were exploited in order to determine the probability that the voxel belongs to feature lines, borders or corner points. Pauly et al. in [32] extended the PCA approach to multiple neighborhood sizes and it enabled to recognize all kinds of features. In [45, 8], Gauss map was built for each data voxel. Voxels were classified by clustering Gauss map. In [21], they extended 2D edge detection filters to 3D filters.

Since mesh-based method requires mesh generation process, its performance depends on the accuracy of mesh generation. Point-based method also has the disadvantage of searching all voxel data, that is, high computational cost. It is also hard to extend most famous methods to detect features on 2D image such as SIFT[17], SURF[4] to 3D image because of much computational costs and memories. Our method uses surface segmentation and mask convolution on voxels. And it is similar to mesh-based method including normal and connectivity information. In addition, the method has low computational cost because it needs filtering operations only on the surface.

3.1.1 Segmentation with the level set method

We employ a segmentation algorithm using the level set method[30, 39]. In image processing, the mathematical model for segmentation was first introduced by Mumford and Shah[27].

For a given image u_0, they decomposed the domain Ω of the u_0 into $\{\Omega_i\}_{i=1,\dots,n}$ such that (a) $\Omega = \Omega_1 \uplus \cdots \uplus \Omega_n$($\uplus$:disjoint union) and (b) each u_i approximates u_0 smoothly on Ω_i by minimizing the following equation:

$$F(\mu, C_0) = \mu L(C_0) + \lambda \int_{\Omega} (u_0 - u)^2 d\Omega + \int_{\Omega/C_0} |\nabla u|^2 d\Omega,$$

where C_0 is the union of the boundaries of the Ω_i and $L(C_0)$ denotes the length of C_0 and both $\lambda > 0$ and $\mu > 0$ are parameters. It is usually assumed that the intensity of the u_0 changes rapidly across C_0. The first term makes the length of C_0 as short as possible, that is, plays the role in reducing noises. The second term fits the image u to u_0 closely. The last term makes the image except edges as smooth as it can. This term also helps denoising.

Chan and Vese[7] proposed a new model based on the above functional but without calculating the gradient. They considered just the case that a given image u_0 is composed of two regions having approximately piecewise-constant intensities, that is, a object and a background. They decomposed the u_0 into two parts by minimizing the following equation:

$$\begin{aligned} F(c_1, c_2, C) = \mu \cdot L(C) &+ \nu \cdot A(inside(C)) \\ &+ \lambda_1 \int_{inside(C)} |u_0(x,y) - c_1|^2 dxdy \\ &+ \lambda_2 \int_{outside(C)} |u_0(x,y) - c_2|^2 dxdy, \end{aligned} \tag{3.1.1}$$

where C is a evolving closed curve in Ω and $A(inside(C))$ means the area of the inside of C and $\mu \geq 0, \nu \geq 0, \lambda_1 > 0, \lambda_2 > 0$ are fixed parameters and the constants c_1, c_2 are the averages of u_0 inside and outside of C respectively. The first and the second term are the regularizers removing noises. The third and the fourth term contribute to fitting the curve C to the boundary C_0 of image as the following way. If the curve C is outside the object, the third term is positive and the fourth term is nearly close to zero. On the other hand, when the curve C is inside the object, the third term is almost zero and the fourth term is positive. If C overlaps with the object partially, both the third and the fourth term are positive. In the end, the fitting terms are minimized when C is on the boundary of the object. If we let the third term and the fourth term $F_1(C)$ and $F_2(C)$ respectively, the above process is described as Figure 3.3.

Song and Chan[39] considered just the three terms in the above Chan and Vese's model (3.1.1):

$$F(c_1, c_2, C) = \mu \cdot L(C) + \lambda_1 \int_{inside(C)} |u_0 - c_1|^2 + \lambda_2 \int_{outside(C)} |u_0 - c_2|^2. \tag{3.1.2}$$

They handled noise with only the length term because both the length term and the area term play the similar role in denoising. In order to apply the level set method to the above functional, they replaced the curve C with a Lipschitz function ϕ. Then they modified the above functional (3.1.2) as

follows:

$$F(H(\phi), c_1, c_2) = \mu(\int_\Omega |\nabla H(\phi))|) \quad + \quad \lambda_1 \int_{inside(C)} |u_0 - c_1|^2 H(\phi) dx \\ + \quad \lambda_2 \int_{outside(C)} |u_0 - c_2|^2 (1 - H(\phi)) dx,$$

where $H(\phi)$ is the Heaviside function of ϕ. Here, they approximated $\int |\nabla H(\phi)| dx$ with

$$\sum_{i,j} \sqrt{(H(\phi_{i+1,j}) - H(\phi_{i,j}))^2 + (H(\phi_{i,j+1}) - H(\phi_{i,j}))^2}$$

where $\phi_{i,j}$ is the value of ϕ at (i, j) pixel. In order to minimize the energy F, they proposed the following algorithm.

First, set the initial value for ϕ. For example, we can set $\phi = 1$ for four boundaries of the (rectangular) image and $\phi = -1$ for the rest. Second, let x be the value of current pixel and c_1, c_2 be the averages for $\phi = 1$ and $\phi = -1$, respectively. Denote m and n as the number of pixels for $\phi = 1$ and $\phi = -1$, respectively. Then if $\phi(x) = 1$, compute the difference between the present and the next energy:

$$\Delta F_{12} = (x - c_2)^2 \frac{n}{n+1} - (x - c_1)^2 \frac{m}{m-1}.$$

If $\Delta F_{12} < 0$, change $\phi(x)$ from 1 to -1. Otherwise, remain $\phi(x)$ unchanged. Similarly when $\phi(x) = -1$, compute

$$\Delta F_{21} = (x - c_1)^2 \frac{m}{m+1} - (x - c_2)^2 \frac{n}{n-1}.$$

If $\Delta F_{21} < 0$, change $\phi(x)$ from -1 to 1, Otherwise, remain $\phi(x)$ unchanged. If denoising is required, include the length term in the above approximation. Repeat the second step until the total energy F does not change.

This algorithm is very fast because its computational time is $O(N)$, where N is the number of input data. It also is able to handling noise. After accomplishing the segmentation of 3D volume image directly with the above algorithm, we can get the binary image of which value inside the object is -1 and value outside the object is 1.

3.1.2 Signed distance function

In the previous section, we have obtained the binary image. For efficiency in later process, we transform the binary image to the signed distance function with the fast sweeping method that we referred in section 2.1.2. With the signed distance function, if we use a threshold value close to 0, we can focus on not the whole image but the surface, that is, the boundary in the binary image. It has the definite advantages in the mask convolution operation for corner or edge detection.

Without the signed distance function, the computational cost of mask convolution is $O(n^3N)$, where $n \times n \times n$ is the size of a mask, N is the total number of voxel in the image. But with the signed distance function, we can reduce the computational cost to $O(n^3N')$, where N' is the number of voxels corresponding to the surface. Usually, since N' can be approximated to $N^{2/3}$, $N' \ll N$. Thus our process with the signed distance function is faster than direct mask convolution on the volume data and the improvement of speed is more than offsetting the cost $O(N)$ of making the signed distance function.

We also use the property of the signed distance function in the edge detection step. Our new-patterned mask was designed with it.

3.1.3 Nonlocal means filtering

Usually, the result obtained from the previous process still has much noises, which cause the detection of false features on the later process. In order to reduce noises in the segmented surface, we considered two methods, where the one was the bilateral filter[42], the other was the nonlocal means filter[12]. When we applied these methods to our segmented result, we could get the better experimental result using the nonlocal means filter, that is, better preserved edges. So we employ nonlocal surface restoration method addressed in [9], which applied the nonlocal means filtering to the implicit surface.

In [9], they calculate the following gradient flow for surface restoration.

$$u_t = \Delta_w u := \frac{1}{2}\mathrm{div}_w(\nabla_w u) = \int_\Omega (u(y) - u(x))w(x,y)dy, \tag{3.1.3}$$

where weight $w(x, y)$ and similarity function $D(x, y)$ are defined by

$$\begin{aligned} w(x,y) &= e^{-|x-y|^2/c_1} e^{-D(x,y)/c_2}, \\ D(x,y) &= \|\phi[x] - \phi[y]\|_2^2, x \in \Sigma_\delta, y \in N_x, \end{aligned}$$

and N_x is a neighborhood of x within Σ_δ and $\phi[x]$ is a 3D patch of ϕ centered at x.

The strategy is as follows. First, it begins with a surface which is the zero level set of a signed distance function ϕ, where the signed distance function ϕ can be acquired with the fast sweeping method[47] from the result in the previous section. Then, calculate the following discretized version of Equation (3.1.3), iteratively:

$$\phi_j^{k+1} = \phi_j^k + dt \sum_{l \in N_j} w_{jl}(\phi_l^k - \phi_j^k),$$

where dt for the CFL restriction is

$$dt = 1/\max_j \{\sum_{l \in N_j} w_{jl}\}.$$

This computation can be focused on the neighborhood of the surface in the implementation because ϕ is the signed distance function. So it doesn't take long. This process stops at some $k = K$th iteration chosen by the user.

3.2 Corner and sharp edge detection

This section presents our corner and sharp edge detection method[25]. From now on, we mean that corner is a point on two-dimensional manifold which has high curvature compared with its neighborhood points and edge is one-dimensional manifold consisting of corners. The method is divided into two main process as shown in Figure 3.5. The corner detection process first obtains the candidates for corners with the idea of the marching cubes algorithm. Then it filters out corners through Sobel-like filtering. For sharp edge detection, we use convolution with new masks. After two processes have finished, the post processing using the idea of SUSAN is adapted to the result in

order to remove false features. The above processes don't require much computational cost because they use not the whole volume data but the surface data resulted from the previous process.

3.2.1 Corner detection

In this step, the marching cubes algorithm is applied to each voxel in the neighborhoods of the segmented surface and the convolution with Sobel-like mask is carried out.

First, voxels obtained from the previous process need to be filtered out by the marching cubes algorithm[18]. There are 15 configurations of marching cubes for polygonization as shown in Figure 3.6 (a). Only two configurations are considered as candidates for corner voxel. That is, cases 1 and 5 in Figure 3.6 (b) are the cubes containing the corner candidate. Once the marching cube algorithm is executed, we have only to search corners from a set of corner candidates instead of searching corners from the whole voxels.

To pick out corners among the candidates, mask convolution with 3D masks in the pattern of Sobel is used, shown in Figure 3.7. While Sobel filter was developed for edge detection originally, we employ it to detect corner. There are three masks Sx, Sy and Sz provided in different directions, i.e. xy, yz, and xz-planes. These masks are convolved with the corner candidates to estimate gradients in those three directions. For a voxel, if its responses, i.e. convolution sum, from filtering through the above masks are large, it is highly likely that the voxel is on edge, that is, plane in 3D space. Thus if its response is less than a threshold, it is marked as corner. Corner can be also considered as voxel having small change in the direction of gradients. If all the ratio of responses are close to 1 or the differences between the ratios is less than a threshold, we mark the voxel as corner. The pseudo-code for corner detection is provided below.

```
Procedure cornerDetect(CC,Sx,Sy,Sz,ts1,ts2)
{
     Gx=Convolution(CC,Sx);
     Gy=Convolution(CC,Sy);
     Gz=Convolution(CC,Sz);
```

```
      Dxy=Gx/Gy ;
      Dyz=Gy/Gz ;
      Dzx=Gz/Gx ;
      For (i,j,k) in CC
      If( (|Dxy(i,j,k)-Dyz(i,j,k)|<ts1 &&
             |Dyz(i,j,k)-Dzx(i,j,k)|<ts1 &&
           |Dzx(i,j,k)-Dxy(i,j,k)|<ts1) ||
           (|Gx(i,j,k)|+|Gy(i,j,k)|+|Gz(i,j,k)|<ts2) )
         (i,j,k) is corner voxel.
}
```

where CC is a set of the corner candidates and both ts1 and ts2 are thresholds. Dxy, Dyz and Dzx mean the directions of the gradients.

3.2.2 Sharp edge detection

In [25], we introduced 3D masks to detect sharp edge voxel. While most existing 3D edge detectors [26, 21, 48, 5] approximate the gradient or Laplacian, we use the property of the signed distance function obtained from the previous process. We employ mask convolution but masks of new patterns. Three masks Cx, Cy and Cz were designed in three different directions, i.e. x-, y- and z-directions, shown in Figure 3.8. Since we adapt these filters to the signed distance function, we can get small responses, i.e. convolution sum, near voxels corresponding to edge. If the response of a voxel is less than some threshold, we mark the voxel as edge. With this, we can detect convex sharp edge.

In addition, other three masks, Cx_i, Cy_i and Cz_i were also designed in order to detect concave sharp edge. Contrary to detection for convex sharp edge, if the response of a voxel is more than some threshold, we denote the voxel as concave edge. The masks of Cy and Cy_i are shown in Figure 3.9. The pseudo-code for detecting sharp edges is provided below.

```
Procedure edgeDetect(B,Cx,Cy,Cz,Cx_i,Cy_i,Cz_i,t1,t2)
{
      Gx=Convolution(B,Cx);
```

```
    Gy=Convolution(B,Cy);
    Gz=Convolution(B,Cz);
    Gx_i=Convolution(B,Cx_i);
    Gy_i=Convolution(B,Cy_i);
    Gz_i=Convolution(B,Cz_i);
    For (i,j,k) in B
        If ( Gx(i,j,k)<t1   || Gy(i,j,k)<t1   ||
             Gz(i,j,k)<t1   || Gx_i(i,j,k)>t2 ||
             Gy_i(i,j,k)>t2 || Gz_i(i,j,k)>t2  )
           (i,j,k) is a sharp edge voxel.
}
```

where B is a set of the boundary voxels corresponding to the segmented surface, t1 and t2 are thresholds.

3.2.3 False feature removal

For the postprocessing, we employ a denoising module. Since both noise voxels and sharp edge voxels involve extreme gradients, we need to distinguish them. The idea of SUSAN method[38] is applied to resolve this. This method starts with a circular window in which the central voxel, so-called nucleus, is the voxel detected as feature(Figure 3.10). The original SUSAN method is used to detect edges or corners and to classify them with the ratio of the interior area over the total area of the circular window, where the interior area means the voxels corresponding to negative values of the signed distance function. For example, the feature can be classified into (i) *salient* if the ratio is less than 0.5, (ii) *flat* if the ratio is approximately 0.5, and (iii) *concave*, otherwise.

But we exploit this idea to remove the false features. When the above criterion is applied to the voxel marked as sharp edge or corner, if the ratio is close to 0.5, it can be treated as noise on the surface of object. In this way, we can reduce salient voxels. For convenience of calculation, we use a cube window to determine total volume and interior volume.

Also, we use inner product between the normal vectors. We judge that if

at least one among the values of inner product between the normal vector at feature voxel and the normal vector at the neighboring voxel is negative, the voxel belongs to the set of features, Otherwise, the voxel is considered as a noise voxel on the surface.

3.3 Results

First, we tested the data with a resolution of $50 \times 50 \times 50$ and the results are shown in Figure 3.11 with different thresholds setting. Red dots represent corner voxels, black dots sharp edge voxels and green dots boundary voxels of the shape. We can check the better result for corner detection in Figure 3.11 (b) than in Figure 3.11 (a).

Figure 3.12 shows the results for the data scanned at a resolution of $300 \times 300 \times 250$. Figure 3.12 (a) shows the visualization of the segmentation result from CT data. It is very noisy. After denoising with nonlocal means filter, we can get the result shown in Figure 3.12 (b). This seems even better because lots of noises have been reduced. But there are still some noises on the surface. Figure 3.12 (c) shows features extracted from (b) without SUSAN method. In this case, we can see some curves, maybe noises, on the surface but not edge. After applying SUSAN method to (c), the result in Figure 3.12 (d) is obtained. The curves corresponding to noises have gone away. Also while part of holes which is not edge definitely was detected in (c), we can check that only edge in holes remains in (d).

We also tested the piston part scanned at a resolution of $291 \times 291 \times 213$. Figure 3.13 shows the result. Like the above example, (c) has much false features because surface is not flat but curvy. But SUSAN method reduces them considerably as shown in (d).

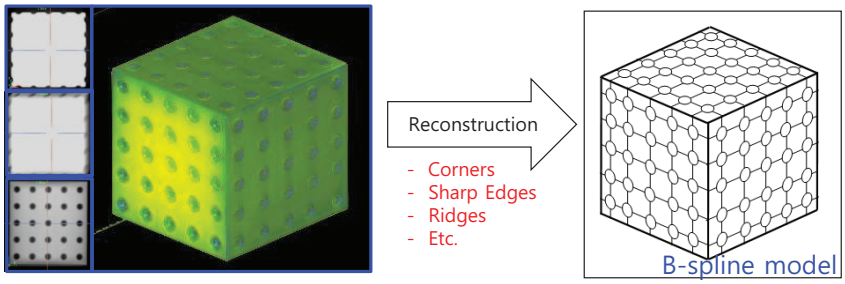


Figure 3.1: Reverse engineering from CT scanned data to B-spline model. In order to get B-spline model, we need to decompose the domain and it requires to detect features such as sharp edge and corner.

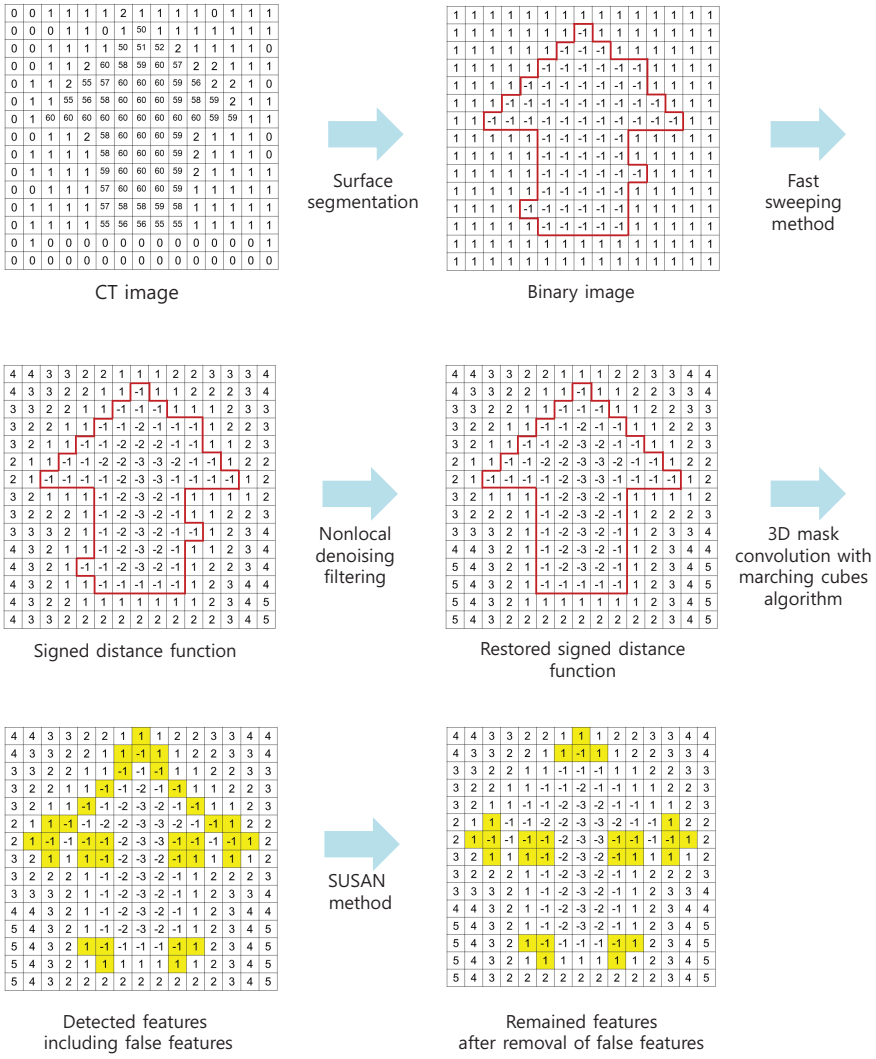


Figure 3.2: Overview of the whole process for feature detection.

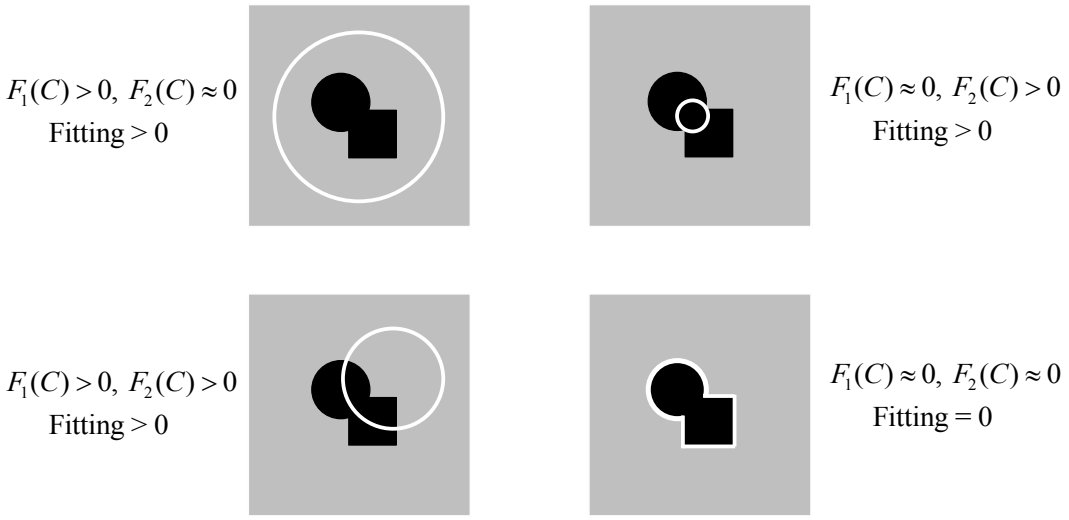


Figure 3.3: The role of fitting term in the Chan and Vese's model. In the above, Fitting means $F_1(C) + F_2(C)$.

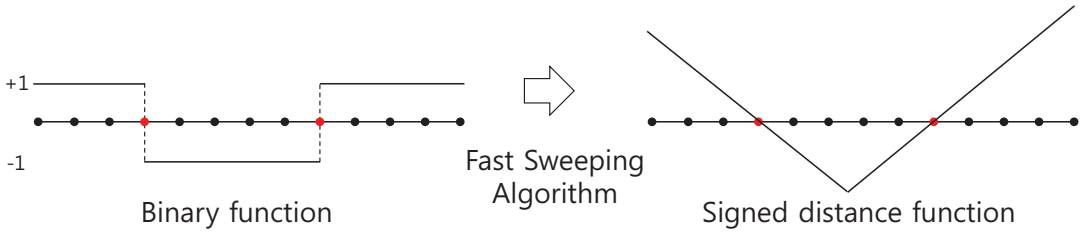


Figure 3.4: Converting the binary image into the signed distance function in one dimension.

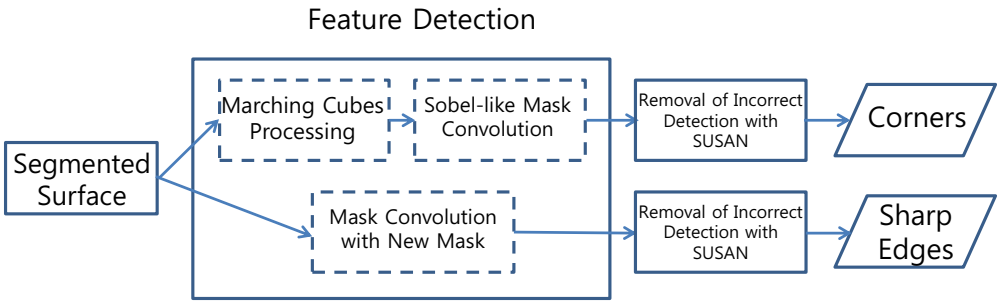


Figure 3.5: The process of corners and sharp edges detection.

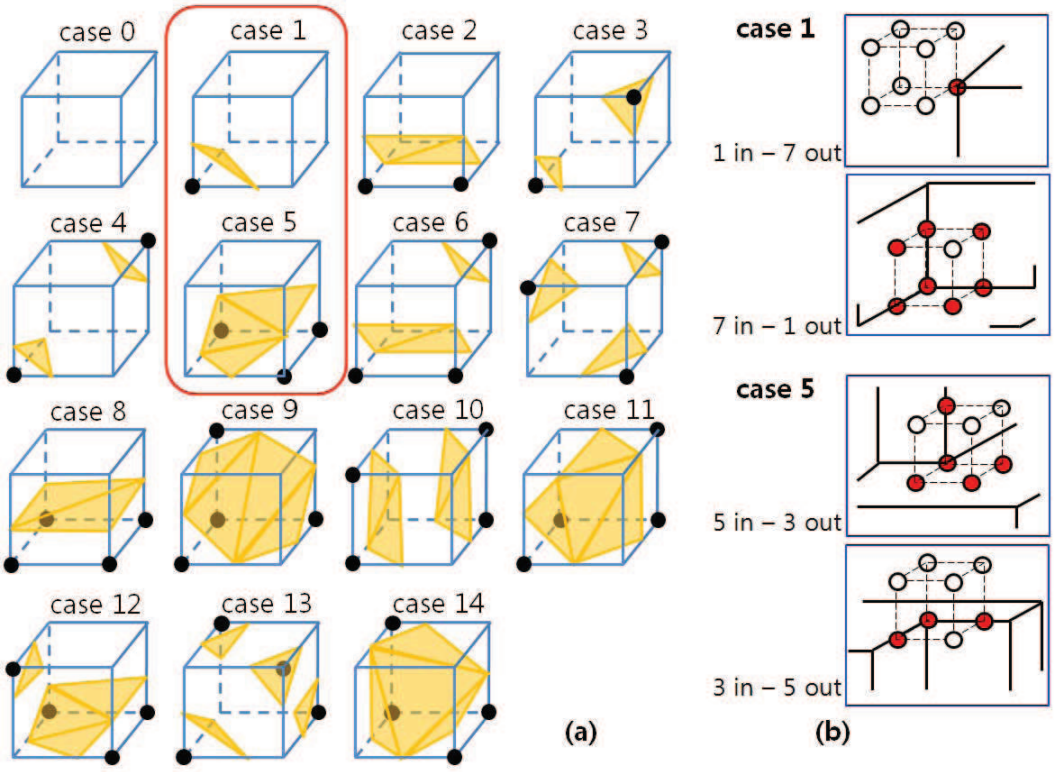


Figure 3.6: (a)Configurations of the marching cubes algorithm. We consider that case 1 and case 5 correspond to corner candidates. (b)Detailed cases for case 1 and case 5.

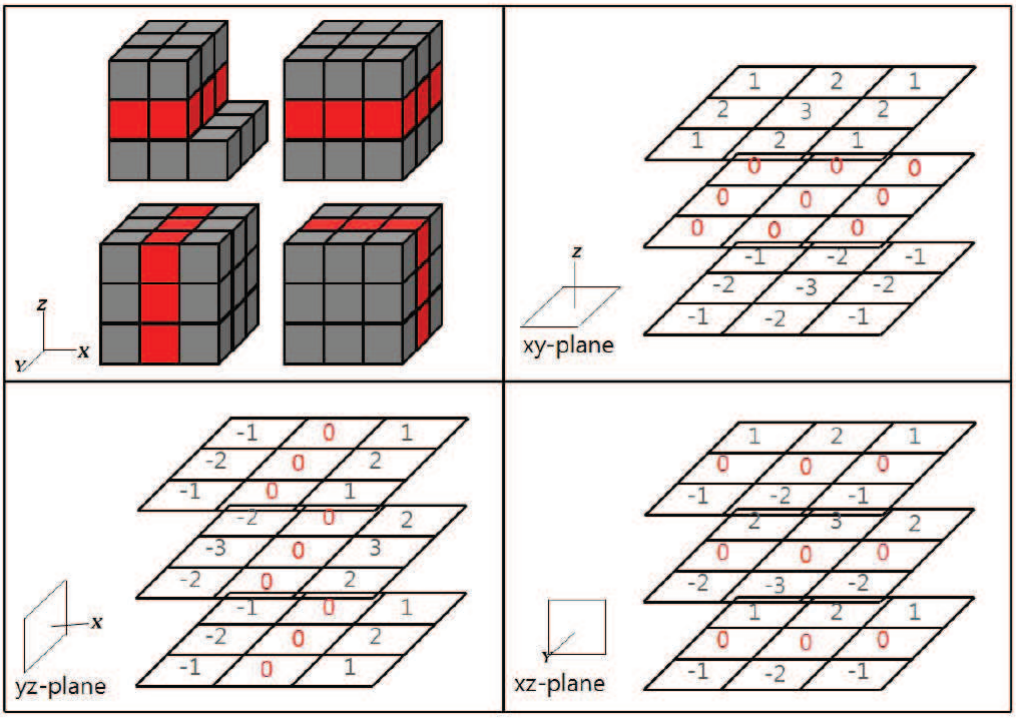


Figure 3.7: Sobel-like pattern of convolution mask for corner detection.

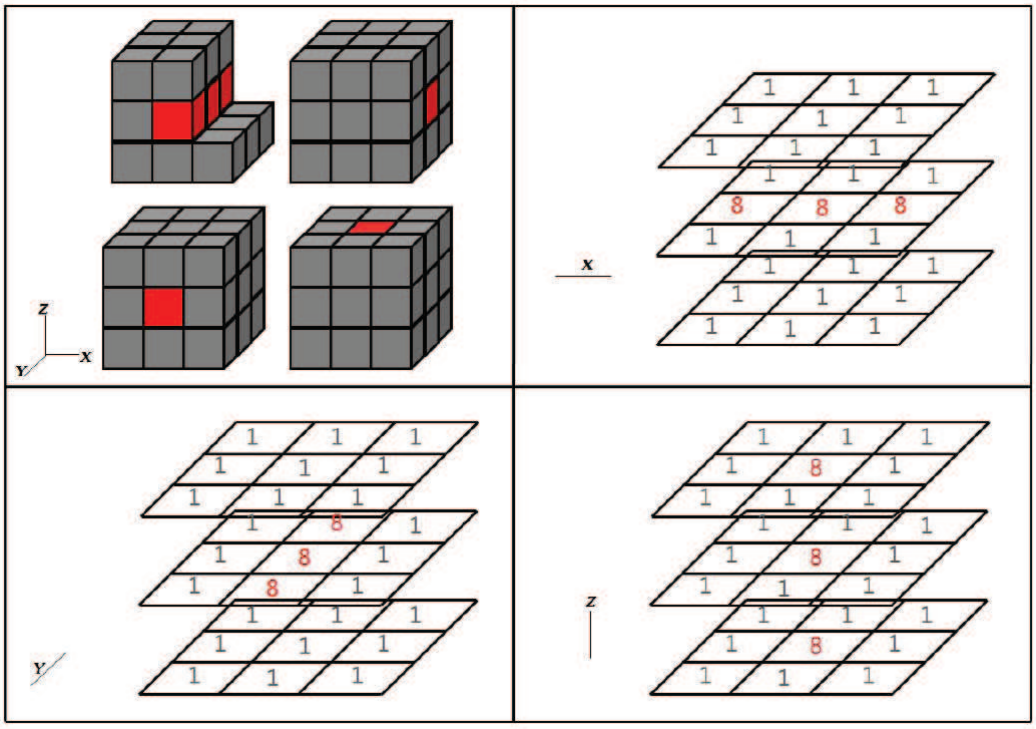


Figure 3.8: Pattern of convolution mask for sharp edge detection.

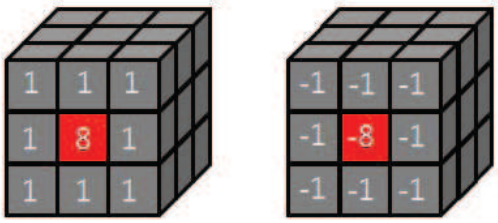


Figure 3.9: Cy mask for convex sharp edge(left), Cy_i mask for concave sharp edge(right).

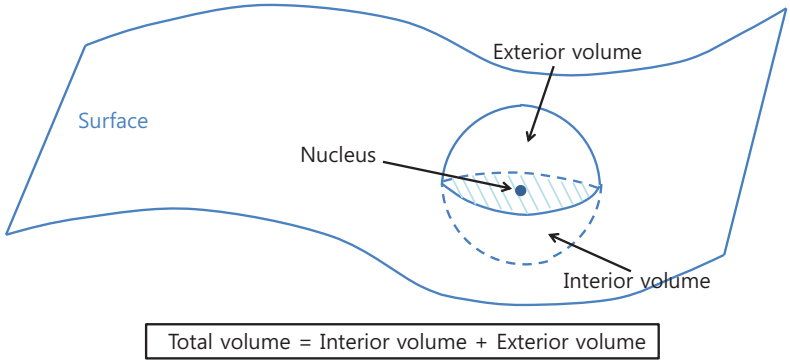


Figure 3.10: A sphere window for determining a SUSAN ratio. The nucleus is characterized according to the volume of sphere.

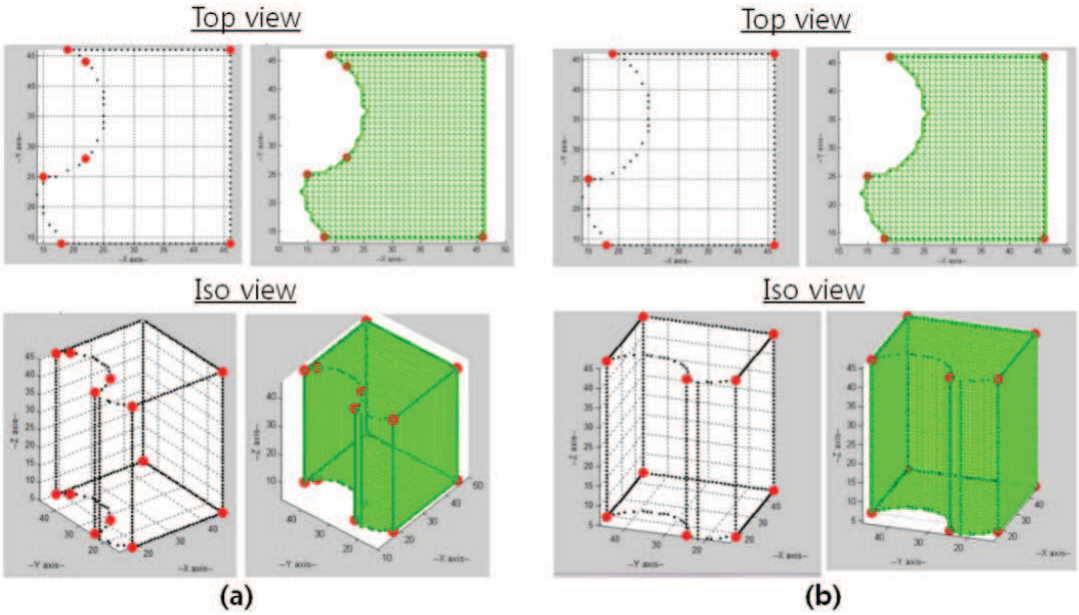


Figure 3.11: (a)thresholds : t1=15, t2=200, ts1=1.2, ts2=500 (b)thresholds : t1=15, t2=100, ts1=0.7, ts2=350. By adjusting threshold, the result (b) better than (a) can be obtained without SUSAN method.

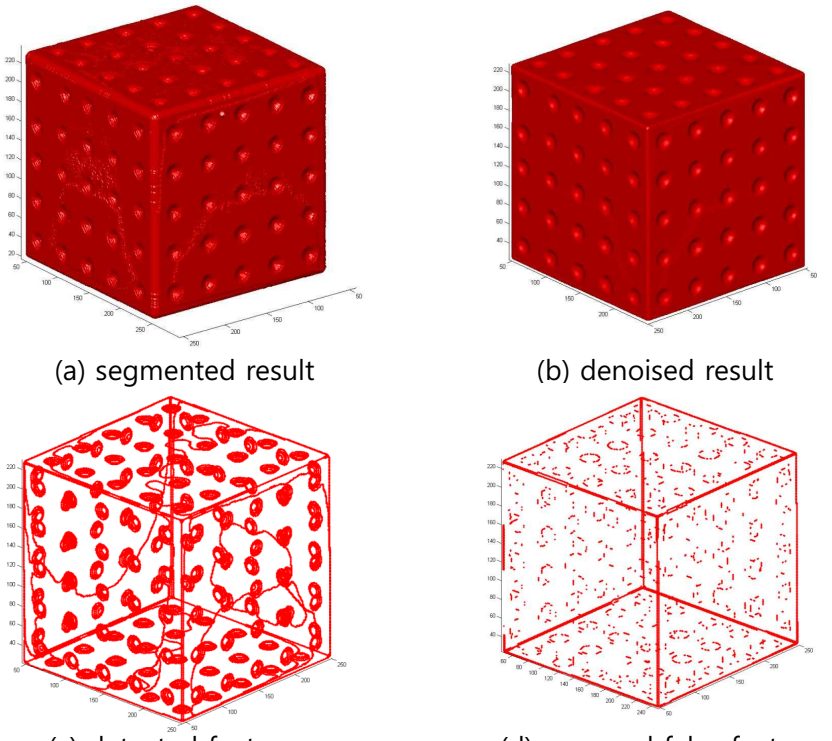


Figure 3.12: Results for cube data scanned at a resolution of $300 \times 300 \times 250$.

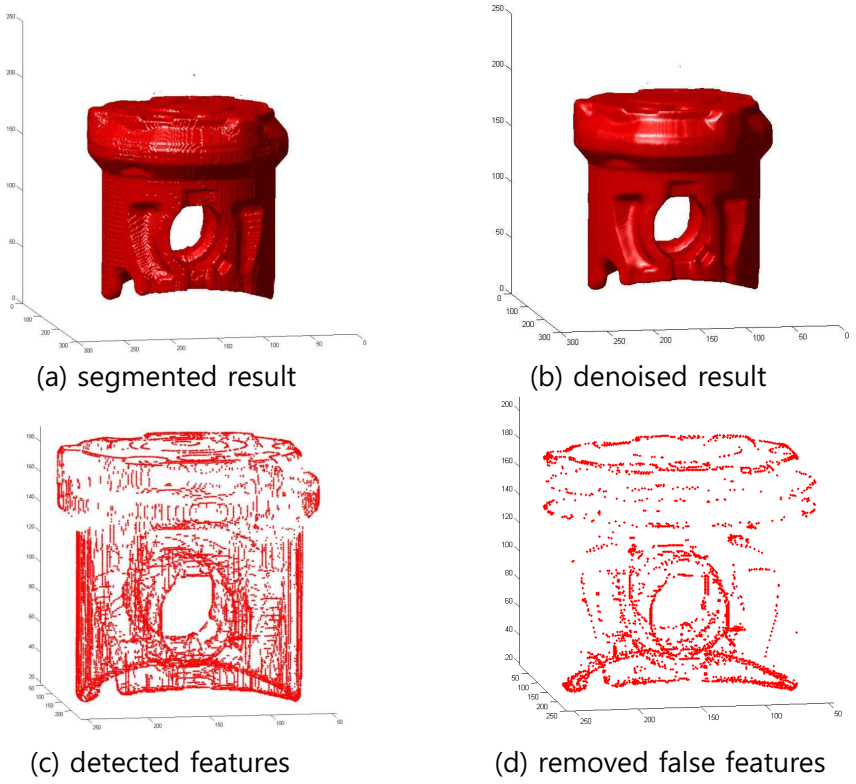


Figure 3.13: Results for piston data scanned at a resolution $291 \times 291 \times 213$.

Chapter 4

Conclusion and Further Work

In chapter 2, we presented a efficient surface reconstruction algorithm using level set method and octree. Although our mathematical model is based on Ye's[46], ours resolves the drawback that Ye's has. Furthermore, our algorithm is more efficient than the previous and can generate the high resolution. We are going to address a new mathematical model which employs not Laplacian term but the term related to curvature for smoothing.

In chapter 3, we introduced a method for corners and sharp edges detection including four main steps. First, the surface segmentation algorithm with level set method is applied in order to extract the surface in form of implicit function from 3D CT scanned volume data. And then the binary image is converted into the singed distance function. This is enable to efficient operation in later process. Nonlocal means filter is used for denoising. Next, as the main step, Sobel-like mask convolution with the marching cubes algorithm and new mask convolution are done to detect corner voxels and sharp edge voxels, respectively. Finally, with the idea of SUSAN method, the features detected incorrectly are eliminated. The computational time of the process with the surface segmentation is faster than mask convolution with the volume data. We are going to develop a method for detecting other features such as ridges. Then we will develop how to connect these resulting features and reconstruct a B-spline model in the end.

Bibliography

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

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

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

[4] H. Bay, T. Tuytelaars and L. V. Gool "SURF: Speeded Up Robust Features", *Proceedings of the 9th European Conference on Computer Vision*, pp. 404-417, 2006.

[5] J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8(6), 1986, pp. 679-698.

[6] 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 Objecets with Radial Basis Functions", *Proceedings of ACM SIGGRAPH*, 2001, pp. 67-76.

[7] T. Chan and L. Vese, "Active Contours without Edges", *IEEE Transactions on Image Processing*, Vol. 10(2), 2001, pp. 266-277.

[8] K. Demarsin, D. Vanderstraeten, T. Volodine and D. Roose, "Detection of closed sharp edges in point clouds using normal estimation and graph theory", *Computer-Aided Design*, Vol. 39(4), 2007, pp. 276-283.

[9] B. Dong, J. Ye, S. Osher and I.D. Dinov, "Level Set Based Nonlocal Surface Restoration", *Multiscale Modeling and Simulation*, Vol. 7(2), 2008, pp. 589-598.

[10] H. Edelsbrunner and E.P. Mücke, "Three-dimensional alpha shapes", *ACM Transactions on Graphics*, Vol. 13(1), 1994, pp. 43-72.

[11] S. Fleishman, D. Cohen-Or, M. Alexa and C. T. Silva, "Progressive Point Set Surfaces", *ACM Transactions on Graphics*, Vol. 22, 2003, pp. 997-1011.

[12] G. Gilboa and S. Osher, "Nonlocal Operators with Applications to Image Processing ", *Multiscale Modeling and Simulation*, Vol. 7(3), 2009, pp. 1005-1028.

[13] S. Gumhold, X. Wang and R. McLeod, "Feature Extraction from Point Clouds", *Proceedings of 10th International Meshing Roundtable*, 2001.

[14] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald and W. Stuetzle, "Surface reconstruction from unorganized points", *Computer Graphics*, Vol. 26(2), 1992, pp. 71-78.

[15] A. Hubeli and M. Gross, "Multiresolution Feature Extraction for Unstructured Meshes", *in: Proceedings of the conference on Visualization '01*, Washington, DC, USA, 2001.

[16] K. Hildebrand, K. Polthier and M. Wardetzky, "Smooth Feature Lines on Surface Meshes", *Proceedings of 3rd Eurographics Symposium on Geometry Processing*, 2005, pp. 85-90.

[17] D. G. Lowe "Object Recognition from Local Scale-Invariant Features", *Proceedings of the 7th International Conference on Computer Vision*, pp. 1150-1157, Kerkyra, Greece, 1999.

BIBLIOGRAPHY

[18] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *Proceeding of SIGGRAPH '87 Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 163-169.

[19] X.-D. Liu, S. Osher and T. Chan, "Weighted essentially non-oscillatory schemes", *Journal of Computational Physics*, Vol. 115(1), 1994, pp. 200-212.

[20] F. Losasso, F. Gibou and R. Fedkiw, "Simulating Water and Smoke with an Octree Data Structure", *Proceedings of ACM SIGGRAPH*, 2004, pp. 457-462.

[21] O. Monga, R. Deriche, G. Malandain and J.P. Cocquerez, "Recursive Filtering and Edge Closing: Two Primary Tools for 3D Edge Detection", *Proceedings of the First European Conference on Computer Vision*, 1990.

[22] O. Monga, R. Deriche and J. Rocchisani, "3D Edge Detection Using Recursive Filtering: Application to Scanner Images", *CVGIP: Image Understanding*, Vol. 53(1), 1991, pp. 76-87.

[23] 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*, Vol. 218, 2006, pp. 123-140.

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

[25] T.-C. Ma, C.-S. Park, K. Suthunyatanakit, M.-J. Oh, T.-W. Kim and M. Kang, "Features Detection from on Industrial Noisy 3D CT Data for Reverse Engineering",*Studies in Computational Intelligence*, Vol. 413, 2012, pp. 89-101.

[26] M. Morgenthaler and A. Rosenfeld, "Multidimensional Edge Detection by Hyper-surface Fitting", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 3(4), 1981, pp. 482-486.

[27] D. Mumford and J. Shah, "Optimal Approximation by Piecewise Smooth Functions and Associated Variational Problems", *Communications on Pure and Applied Mathematics of Computation*, Vol. 42(5), 1989, pp. 577-685.

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

[29] S. Osher and R. Fedkiw, "Level Set Method and Dynamic Implicit Surfaces", *Springer*, 2003.

[30] S. Osher and J. Sethian, "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on a Hamilton-Jacobi Formulations", *Journal of Computational Physics*, Vol. 79, 1988, pp. 12-49.

[31] S. Popinet, "Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries", *Journal of Computational Physics*, Vol. 190, 2003, pp. 572-600.

[32] M. Pauly, R. Keiser and M. Gross, "Multi-scale Feature Extraction on Point Sampled Surfaces", *Computer Graphics Forum*, Vol. 22, 2003, pp. 281-289.

[33] C. Park, C. Min and M. Kang, "Surface Reconstruction of Scattered Point Data on Octree", *Journal of Korean Society for Industrial and Applied Mathematics*, Vol. 16(1), 2012, pp. 31-49.

[34] V. Raja and K. J. Fernandes, "Reverse Engineering: An Industrial Perspective", *Springer*, 2007.

[35] Y. Saad, "Iterative Methods for Sparse Linear Systems", *University of Minnesota*, 2003.

[36] P. Sonneveld, "CGS, A fast Lanczos-type solver for nonsymmetric linear systems", *SIAM Journal on Scientific and Statistical Computing*, Vol. 10, pp. 36-52, 1989.

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

[38] S.M. Smith and J.M. Brady, "SUSAN - A New Approach to Low Level Image Processing", *International Journal of Computer Vision*, Vol. 23(1), 1997, pp. 45-78.

[39] B. Song and T. Chan, "A Fast Algorithm for Level Set Based Optimization", *CAM-Report 02-68*, 2002.

[40] C.-W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock capturing schemes", *Journal of Computational Physics*, Vol. 77, 1988, pp. 439-471.

[41] J. Strain, "Fast tree-based redistancing for level set computations", *J. Comput. Phys.*, Vol. 152, 1999, pp. 664-686.

[42] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images", *Proceedings of Computer Vision, 1998, Sixth International Conference*, pp. 839-846, Bombay, India, 1998.

[43] H. A. van der Vorst "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems", *SIAM Journal on Scientific and Statistical Computing*, Vol. 13, pp. 631-644, 1992.

[44] K. Watanabe and A.G. Belyaev, "Detection of Salient Curvature Features on Polygonal Surfaces", *EUROGRAPHICS 2001*, Vol. 20(3), 2001, pp. 385-392.

[45] C. Weber, S. Hahmann and H. Hagen, "Methods for Feature Detection in Point Clouds", *Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)*, Vol. 19, 2011, pp. 90-99.

[46] 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.

[47] H. Zhao, "A FAST SWEEPING METHOD FOR EIKONAL EQUATIONS", *Mathematics of Computation*, Vol. 74, 2004, pp. 603-627.

[48] S.W. Zucker and R.A. Hummed, "A Three Dimensional Edge Operator", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 3(3), 1981, pp. 324-331.

[49] H. Zhao, S. Osher, B. Merriman and M. Kang, "Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method", *Comput. Vis. Image Underst.*, Vol. 80, 2000, pp. 295-314.

[50] 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)*, Vol. 2, pp. 194-201, Washington, DC, USA, 2001.

국문초록

본 논문에서는 레벨셋으로 표현되는 음함수 곡면을 이용한 역공학 과정을 연구하였다. 우리는 두가지 방법을 고려한다. 첫번째는 팔진트리상에서 산재한 점군으로부터 음함수 곡면을 재구성하는 방법이고 두번째는 구성된 음함수 곡면에서 엣지와 코너와 같은 특징점들을 탐지하는 방법이다.

우리의 곡면 재구성 방법은 팔진트리를 이용하는 레벨셋 방법에 기반한다. 우선 우리는 이전 방법들과 달리 타원형 편미분 방정식 모델을 이용한 곡면 재생성 방법을 고려한다. 이 방법은 2010년에 Ye에 의해 제안되었으나 균일 격자상에서 구현될 경우 많은 메모리를 소비하는 단점이 있었다. 우리는 팔진트리 구조상에서 이 모델을 구현함으로써 그러한 단점을 해결하였다. 이 과정에서 새로운 거리함수 계산 방법이 도입되었다.

우리는 또한 3차원 CT 영상으로부터 특징점들 탐지하는 방법을 연구하였다. 레이저 스캐너는 정확도가 높고 잡음이 적은 반면에 물체의 내부를 조사할 수 없다. 이러한 이유로 물체의 내부까지 찍을 수 있는 CT 스캐너가 기계부품의 비파괴 검사에 이용되는 등 인기를 끌고 있다. 하지만 역공학 과정에서 3차원 영상 데이터는 CAD 소프트웨어에서 사용될 수 있는 B-스플라인 곡면 데이터로 변형되어야 한다. 즉, 음함수 곡면에서 매개변수 곡면으로 변형되어야 한다. 이 과정에서 곡면 매개화를 위해 특징점들을 탐지하는 것이 필요하다. CT 영상은 레이저 스캔 영상보다 많은 잡음을 갖는 단점이 있다. 우리는 이러한 잡음을 줄이는 알고리즘을 전처리과정에서 CT 영상 데이터에 적용한 뒤 우리의 특징감지 방법을 사용하여 특징점들을 탐지한다.

주요어휘: 레벨셋 방법, 곡면 재구성, 음함수 곡면, 편미분방정식, 팔진트리, 특징탐지, 역공학
학번: 2007-30763

감사의 글

박사 입학할 당시만 하더라도 나에게 이런 날이 올까 하는 막연한 느낌밖에 들지 않았다. 그러던 내가 지금 졸업논문을 마무리하면서 이글을 쓰고 있다니 감회가 새롭다. 힘들다면 힘들 수 있는 대학원 생활을 나름 재미있게 보낼 수 있었던 건 주위의 고마운 분들 때문이었던 것 같다. 물론 가장 고마운 분은 항상 나에게 무한한 신뢰를 보내주시고 의지가 되는 부모님이다. 언제나 제 선택을 존중해 주시고 묵묵히 지켜봐 주셔서 감사합니다. 사랑합니다.

못난 제자가 졸업할 수 있도록 많은 기회를 주시고 기다려주신 강명주 교수님 정말 감사합니다. 졸업 후에도 열심히 해서 교수님 명성에 누를 끼치지 않도록 하겠습니다. 프로젝트 경험을 통해 저에게 많은 것을 가르쳐 주셨던 김태완 교수님께도 감사드립니다. 저에게 해주셨던 많은 좋은 말씀 잊지 않겠습니다. 공동연구를 하면서 저에게 학문적인 것 뿐만 아니라 학문 외적으로도 많은 노하우를 전해주셨던 민조홍 교수님께도 깊이 감사드립니다. 제가 인간적으로 너무 좋아하는 홍병우 교수님께는 고맙기도 하고 죄송하기도 합니다. 공동연구를 통해 뭔가 좋은 결과가 있었으면 좋았을텐데 제가 기대에 못 미쳐 죄송할 따름입니다. 다음에 꼭 기회가 된다면 제가 도움이 되도록 열심히 하겠습니다. 그리고 제 졸업논문 심사에 흔쾌히 응해주신 하승열 교수님과 박사논문자격시험 심사를 해주셨던 이기암 교수님께도 감사드립니다.

팀에 들어와서 6년 반 동안 MT도 다니면서 재미있게 지냈던 지훈이형, 승현이형, 승미 누나, 선희, 지은이, 명민이 모두 고맙고 앞으로 하는 일 다 잘 되길 빌게요. 특히 지훈이형은 제가 여러가지로 많이 배웠어요. 감사합니다. 승현이형, 승미 누나도 졸업 축하해요. 병준이, 성주, 둘투야, 미소, 승훈이, 고광, 한울이, 경민이, 달현이, 미림이, 준기, 현욱이, 희재, 진의 모두 열심히 해서 다들 무사히 졸업하길 바란다. 한결이, 연후도 졸업 축하하고 좋은 직장 얻었으면 좋겠다. 정미연 박사님, 이호준 박사님, 우현균 박사님도 원하는 바대로 잘 풀리셨으면 좋겠습니다.

대학원 들어와서 가장 친하고 재미있게 지냈던 친구들 의용이, 호, 하균이 모두 고맙다. 의용이랑 호는 좋은 직장 구해서 너무 기쁘고 축하한다. 의용이는 결혼도 축하한다. 하균이도 무엇을 하든 네가 원하는대로 잘 풀리기 빈다. 성훈이랑 승진이도 교수 임용 축하한다. 마음 잘 맞는 대학원

동기 이혁이, 같은 연구실이었던 은주, 옆 연구실의 민하씨, 라미 누나, 형성이, 수정씨, 그리고 그 옆 연구실의 문진씨, 성은씨, 용덕씨, 수다 친구가 되어 줘서 고마웠습니다. 지금은 졸업하신 명선이형, 형태, 요시키. 덕분에 지난 학기 재미있게 보냈어요. 석사 때 단짝이었던 재광이도 너무 고맙고 성욱이, (소)형석이, 용진이, 선정이, 광신이, 명호, 지혜, 민재, (신)형석이, 은희씨, 윤주, 상욱이, 성원씨, 명훈이형, 희철이형, 영호, 창균이, 남균이, 한봄이, 재협이, 선호, 근우형 모두 잘 되서 좋은 소식만 들렸으면 좋겠습니다. 항상 친절하게 대해주시는 김임범 박사님과 남현 박사님께도 감사드립니다. 과사무실 갈 때마다 반갑게 맞아주시는 이국현 실장님, 정철호 실장님, 그외에 과사무실 직원 분들과 수학연구소 직원분들 그 동안 정말 감사했습니다. 학부친구이자 한때 대학원 친구였던 치홍이도 울산가서 잘 적응하는 모습보니 좋고 되도록 빨리 졸업했으면 좋겠다.

마지막으로 제가 대학원 들어와서 몇 번의 슬럼프가 있었는데 그 때마다 극복하는데 도움을 받았던 신신약국 박원장님, 턱관절 치료를 해주신 다니엘치과 김형준 원장님께 감사드립니다.

이 분들 외에도 저와 한번 이상 스친 인연을 갖고 계신 모든 분들께 감사드립니다. 서로에게 좋은 향기만 남았기를 기원합니다.