平成 21 年度 博士論文

Processing of Scanned Geometry Using Spherically Supported Functions
（球被覆関数によるスキャン形状処理）

指導教員　鈴木 宏正 教授

東京大学大学院 工学系研究科 先端学際工学専攻

37-077163

NAGAI, Yukie

長井 超慧

Abstract

Shape representation with spherically supported functions attracts attention in geometry processing recently because of its strong flexibility and ability of representing complex objects. Some geometry processings use differential operators however, spherically supported functions have no proper differential operator.

This thesis derives the discrete differential operators on a set of spherically supported functions: the gradient, divergence and Laplacian operators. This thesis develops also two instances of improved ability of representation of spherically supported functions with discrete differential operators: algorithm for surface reconstruction from a set of points and skeletal structure extraction from a volumetric data. For surface reconstruction, smoothing the spherically supported functions using the discrete differential operators achieves smooth and precise surfaces even from heavily noisy sampling points. In order to realize outlier-robustness, applying Graph-cut before smoothing is proposed. In addition, a useful measure of reliability of a sampling point is proposed.

The proposed algorithm for skeletal structure extraction is based on an observation that the skeletal structure is very close to the maxima of intensity. In this research, intensity is approximated with spherically supported functions. The evaluation of the maxima of the approximated intensity is performed utilizing the discrete differential operators. As a more noise-robust extension based on the smoothing of the spherically supported functions are proposed.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

In recent years, geometry processing obtains higher abilities to represent shapes of 3D real objects as the rapid developing of scanning devices as laser scanners, CT-scanning methods, MRI and so on. Fig. 1.1 and Fig. 1.2 show examples of models scanned using a laser scanner and an X-ray CT scanner respectively.

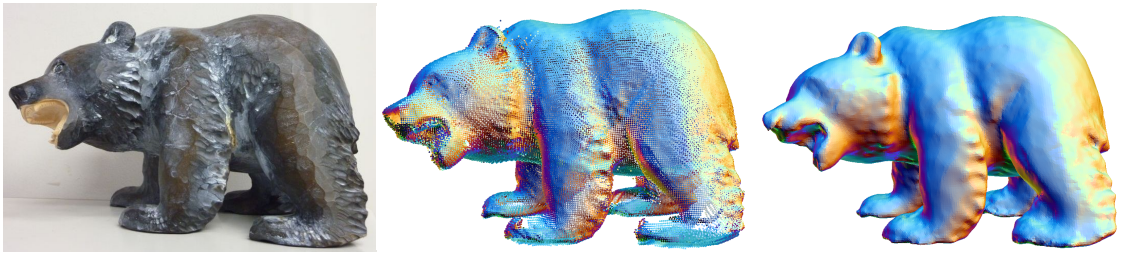


Fig. 1.1. Shape acquisition with a laser scanner.
Left: original 3D object. Middle: scanned data, a set of points. Right: mesh representation.

Fig. 1.2. Shape acquisition with an X-ray CT scanner.
Left: original 3D object. Middle: a horizontal slice of CT-scanned image. Right: isosurface of CT-scanned image.
The left and the center images are courtesy of the Stanford Computer Graphics Laboratory.

Geometry processing shows progress in many attractive applications such as visualization, physical analysis, and processing shapes. These applications play important roles in wide areas involving computer graphics, industrial design, reverse engineering, medical engineering, and so on. In such applications, shapes are generally represented using meshes. As an example in Fig. 1.3, mesh is a shape representation defined by the combination of positions and connectivity of points. Currently many applications adopt mesh as a standard shape representation.

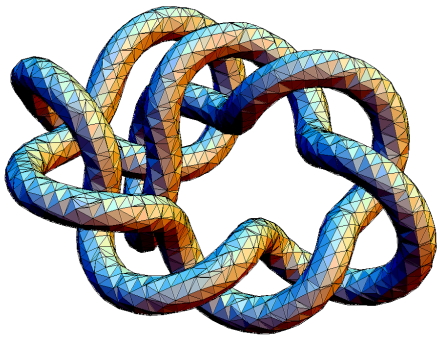


Fig. 1.3. Mesh of a complicated object.

Although scanning technology achieves high-level of accuracy than before, shape reconstruction methods require higher quality scanned data to achieve quality reconstruction. If the quality of scanned data are not enough, reconstructed shapes may be drastically different from the original shape. Fig. 1.4 shows examples of low-quality scanned data and reconstructed mesh. Low-quality data is data which includes some problematic parts as noise and gaps, details in Chap. 2.1.1. From such data, meshes with some artifacts may be generated as the image on the right. It has some undesired surfaces extending forward from the head, the bird on the girl's hand, and her knee where the scanned points are wrongly aligned or not obtained as shown in the center image. Of course wrongly reconstructed surfaces should not be used in applications because it is clear that they will cause errors in geometry operations.

In these days, another shape representation which uses approximation functions of a set of points gets a lot of attention. This is a representation, for points obtained from a surface, expressing the shape of the object with a iso-contour of an scalar field defined through a function which is generated using scanned points. Or, for volumetric data, this method approximates the attribute values (e.g. intensities) equipped with each points with values of an approximating function.

Fig. 1.5 shows an example of implicit representation for a surface-scanned data. This shows a result of reconstruction for points scanned from a surface of a 2D object. The

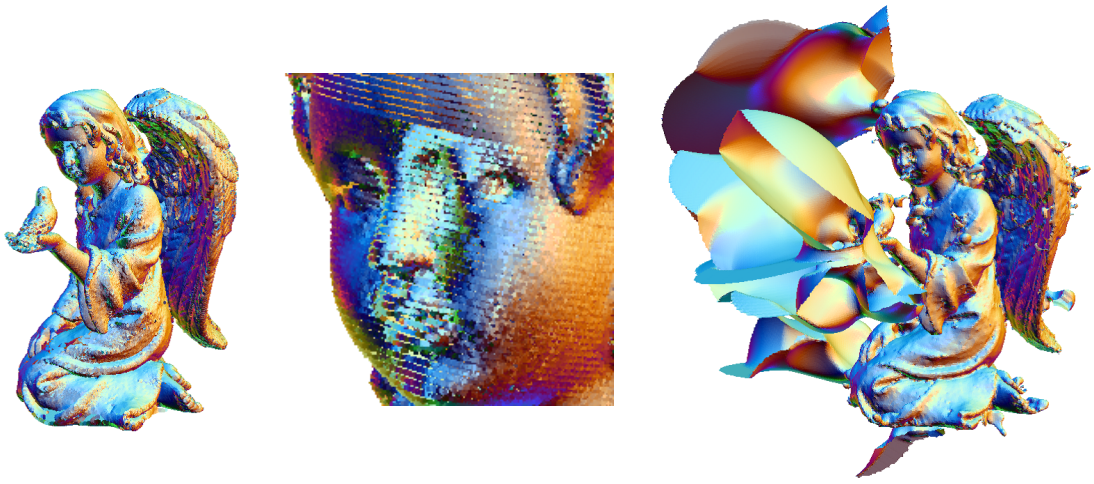


Fig. 1.4. Mesh generated from low-quality data.
Left: low-quality scanned data.
Center: close-up of the face part of the left image.
Right: mesh generated from the points shown in the left image.

left image shows the set of scanned points. The scalar field obtained from these points are indicated in the right image. Color indicates a high value with red and low with blue. The iso-contour representing the object's shape is drawn with a black line.

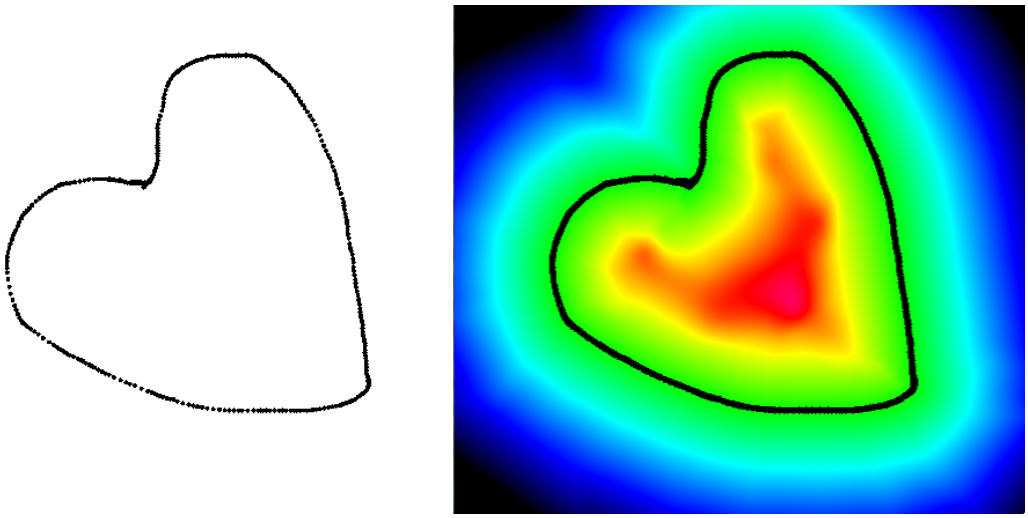


Fig. 1.5. Implicit shape representation.
Left: points scanned from an object. Right: a scalar field and an iso-contour (a black line) representing the object's shape.

Among many methods to generate scalar fields, we especially focus on a technique covering a space which includes scanned points with a set of spheres equipped with local functions. The supports of the local functions are corresponding spheres themselves. In this method, an approximating function is obtained with blending the local functions. Fig. 1.6 is a set of spheres for the points from a surface in Fig. 1.5. Spherical supports and

those centers are shown with pink circles and blue points respectively.

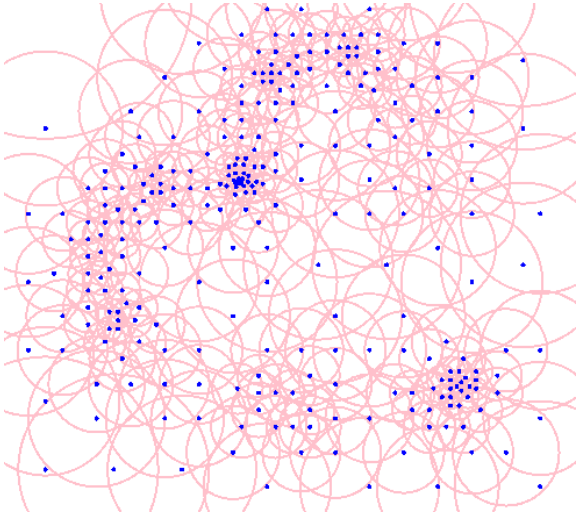


Fig. 1.6. Spherical cover for the scanned points in the left image of Fig. 1.5.

Spherically supported functions have high abilities to shape representation: smooth surfaces, sharp features (edges and corners, see Fig. 1.7). It can approximate attributes of volume models. Moreover, shape representation using spherically supported functions is also very suitable for the operations that is hard for mesh representation. Some operations as blending and set operation are difficult to be operated on mesh, but they can easily operated for models represented with spherically supported functions.

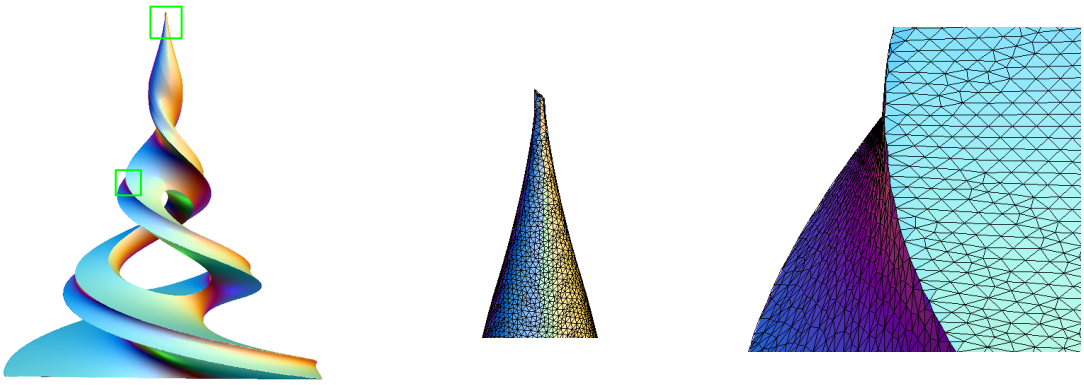


Fig. 1.7. Sharp features. Left: corner. Right: edge.

As mentioned above, the de-facto standard for shape representation in some application areas is mesh, and if needed, models represented with spherically supported functions can be easily converted to meshes with the aid of a method to polygonize an isosurface. The famous methods are Marching Cubes [LC87] and dual contouring [JLSW02]. The iso-contour in Fig. 1.5 is visualized using mesh generated with such a polygonization method.

Geometry processings in applications often encounter difficulties. Some of such process-

ings, for instance blending and set operations mentioned above, can be effectively handled using representation with spherically supported functions rather than mesh. These advantages have been recognized in several application areas, and spherically supported functions are gradually being adopted.

As stated above, spherically supported functions are excellent method as long as these local approximations represent an original shape well. Sometimes good approximations cannot be available because of problems of scanned data. We overcome these situations through smoothing local approximation functions. Smoothing requires smoothing operator (the Laplacian). Unfortunately, there is no proper differential operators on spherically supported functions. This is the motivation of our derivation of differential operators. In this thesis, we enhance the spherically supported functions through deriving differential operators which are discretized in terms of spheres.

Again from the point of applications, until now, because of the absence of proper discrete differential operators, there are some processings that may not be handled well with spherically supported functions. In these cases, utilizing the traditional and continuous differential operators is one solution. Now our discrete differential operators have advantages over such alternative solutions. In order to realize these advantages, we also developed two application algorithms: surface reconstruction and skeletal mesh extraction.

1.2 Problem Setting

Deriving Differential Operators. Differential operators are essential operators for every representation in order to analyze geometrical characteristics of objects. Unfortunately, spherically supported functions had no unique differential operators. So deriving differential operators specified for a set of spherically supported functions is worthful. The fact that required final models are generally mesh indicates that assuming supported functions are linear is natural. So we derive the differential operators on a set of spherically supported functions in the case these are linear functions.

Surface Reconstruction. Surface reconstruction is a process generating a mesh from a set of points. It is much in demand in many application areas so a number of algorithms are proposed. Some explicit methods involving Delaunay based methods such as crust [ABK98] and Tight cocone [DG03] adopt input points as vertices of mesh, so generated mesh is affected by the quality of input points. Generally input points are scanned data which may include some problems such as much amount of noise and lack of sampling. On the other hand, shape representation with an implicit method which approximates input points can

counteract to problems of scanned data. Ability to treat scanned data directly is a great advantage of implicit methods while explicit methods may need preprocessing as denoising; cleaning scanned data through removing noises and differences of sampling densities. Among of implicit methods, spherically supported functions can represent details well. So our aim is generate a approximated shape of an object robustly to low-quality data on the basis of implicit method.

As denoted in the previous section, shape representation with spherically supported functions is different from mesh. If applications need mesh, obtained shape models can be converted into meshes via a polygonizing method.

Skeletal Structure Extraction. Skeletal structure is a useful structure to understand intuitively the shape of an object. There are many kinds of representation of skeletal structure, for instance, a set of voxels, 1D curve, 2D sheet, and mixture of them. In this thesis, we especially focus on 2D sheet-like structure. See Fig. 1.8, an example of a 2D skeletal sheet for an industrial object.

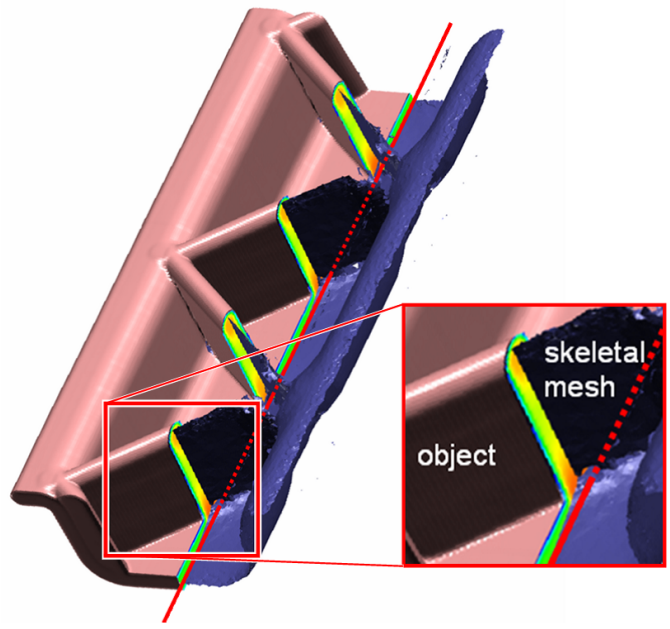


Fig. 1.8. Industrial object and its skeletal sheet.

Major methods for extracting such a skeletal structure are Voronoi based methods. They take surfaces of objects as input. But in many cases, data of industrial objects are obtained as volumetrically scanned data. So algorithm which can directly deal with volumetric data are useful. Since scanned data typically has problems involving noise, algorithm is demanded to generate smooth skeletal structure from scanned data.

1.3 Contributions

In this thesis, the discrete differential operators on spherically covered functions are proposed. In addition, as applications of spherically covered functions extended with differential operators, algorithms for surface reconstruction and extraction of 2D skeletal structure are presented.

These two application algorithms achieve noise robustness and generation of smooth structures.

1.3.1 Differential Operators on Spherically Covered Functions

For the existing spherically supported functions, we derived discrete differential operators [NOS09]: the divergence and Laplacian operators of a vector field, and the Laplacian operator of a scalar field. They are defined on the centers of supports where various estimations are performed. In the derivation of the operators, three conditions are assumed: spherically supported functions are linear, each support has a constant vector, and boundary of each support are discretely approximated with intersections with incident supports.

Proposed operators are very fundamental, so they can be applied into various application areas where differential operators are useful. Applications in surface reconstruction and skeletal mesh extractions are proposed in this thesis.

1.3.2 Surface Reconstruction

An algorithm to approximate the surface of an object from points on the surface of the object is developed on the basis of extended spherically supported functions. Generated surface is converted to a triangular surface mesh with the aid of an existing polygonization method. This algorithm has an advantage to handling a data with outliers, lack of samplings. Generated surface is smooth and represents fine features of an object.

These advantages are achieved by combination of global and local approaches: spherically covered functions and Graph-cut.

Moreover, smoothing extended spherically supported functions using differential operators defined in this thesis is also combined. This smoothing achieves noise-robustness and more smooth representation. In the smoothing process, the way to make smoothing anisotropic is developed. This method enable smooth representation and preservation of highly-curved regions. Moreover, a measure of confidence of points are proposed.

1.3.3 Skeletal Structure Extraction

An algorithm to generate a surface mesh representing the 2D sheet-like skeletal structure of an object directly from volumetric data such as CT-scanned data is developed. In our interest, target object consists of one material thin plates as structures typically found in industrial engineering.

Our approach is based on the observation that the maxima of intensity and skeletal structure of our target object are resembled each other. First, the intensity of scanned data is approximated with spherically supported functions, then the maxima of intensity are estimated. As a result of approximation approach, noise-robustness and smooth skeletal structure generation are achieved. Since using the spherical supports as an intermediate structure, an adaptive surface mesh can be generated. Moreover, obtained skeletal structure reaches to the boundary of objects. This characteristic is suitable for some industrial applications.

In addition, to improve the robustness in maxima estimation, we propose to replace the spherically supported functions with extended ones, and smooth the approximation functions with the discrete differential operators.

1.4 Thesis Organization

First in Chapter 2, basic knowledge about shape representation are denoted. Then previous works related to the subjects dealt with in this thesis are introduced. Next, the definition of spherical supported functions and generation methods of them are explained in Chapter 3. And then Chapter 4 derives the differential operators on a set of spherical supported functions. The following chapters are devoted to applications of spherical supported functions enhanced with discrete differential operators. Chapter 5 provides an algorithm for surface reconstruction from points scanned from surfaces. This algorithm includes smoothing process realized with the discrete differential operators. Chapter 6 introduces a skeletal structure generation from volumetric data. In order to further improve skeletal structure's quality, applying the discrete differential operators is proposed. Finally Chapter 7 concludes this thesis and suggests future directions. Fig. 1.9 visualizes the organization of this thesis.

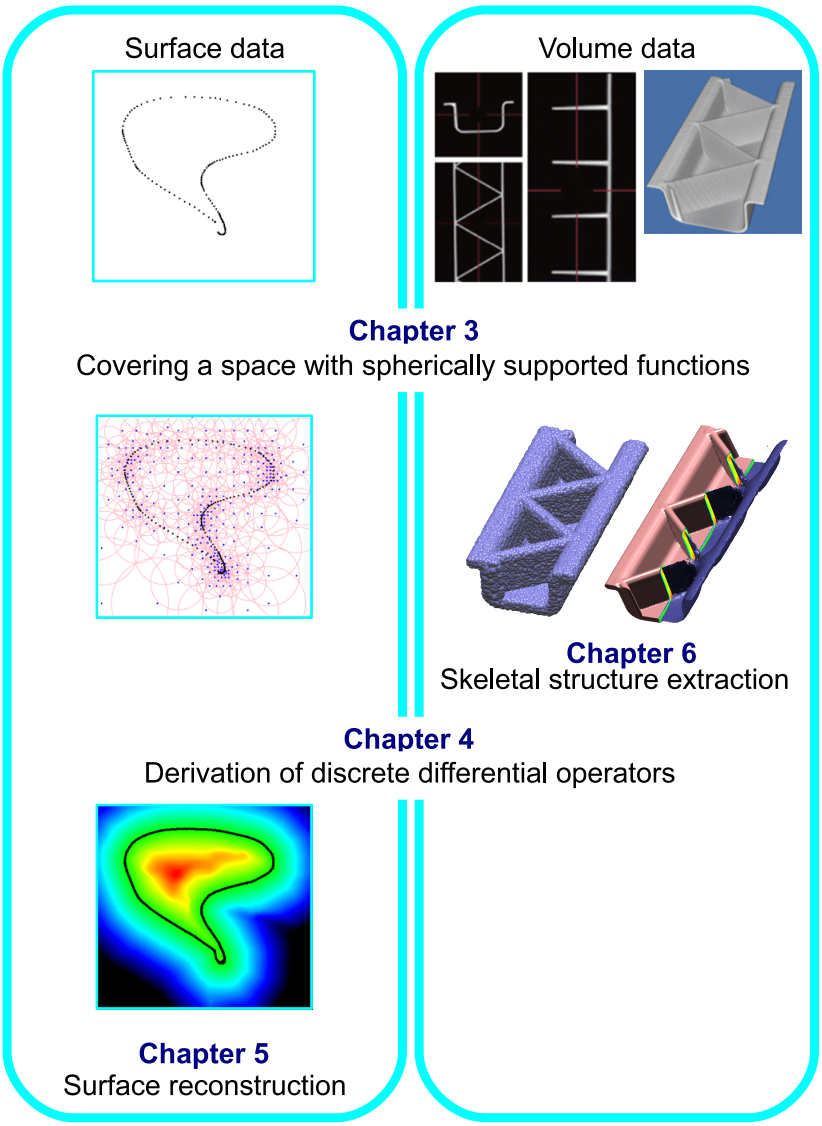


Fig. 1.9. The organization of this thesis.

Chapter 2

Background and Related Work

In this chapter, fundamental knowledge about shape representation which we concern with throughout this thesis is provided in the first section. In the next section, previous works related to the topics discussed in this thesis are introduced.

2.1 Shape Representation

In this section, fundamental knowledge about measured data is described. It is followed by an introduction of mesh: the most common shape representation for scanned data in areas such as computer graphics and industrial engineering.

2.1.1 Measured Data

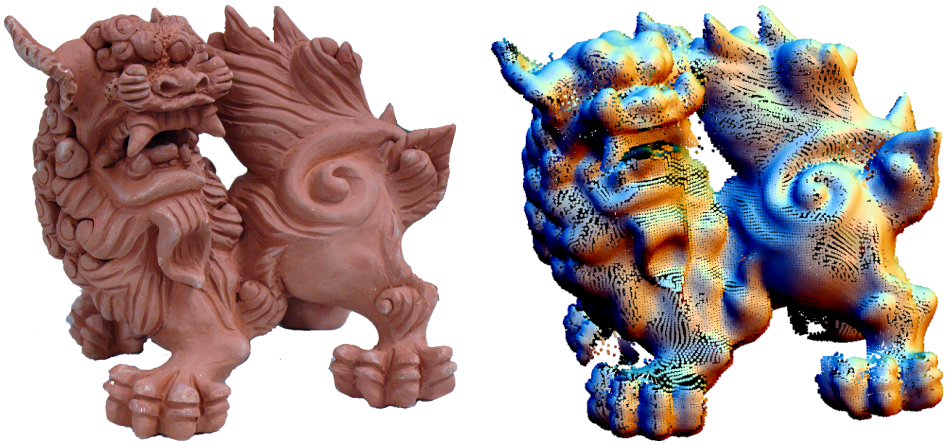


Fig. 2.1. Real object and scanned data.

Recently, as a method to obtain a shape of a real 3D object, measuring the target object directly using scanning devices is a common way. Laser scanners, x-ray CT scanners and MRI scanners are examples of such devices. Measured points obtained with a measuring method are generally referred to as **sampling points**.

In the case only shape of a target object is needed, it is enough to scan the surface

of the object. In this case, using scanning devices such as a laser scanner is a major solution. Scanned data obtained with a laser scanner consists of a set of sampling points from surface of the object. A set of sampling points is specially called as a **point cloud**. Each point of the point cloud $\{\boldsymbol{p}_i\}$ has its coordinate

$$\{\boldsymbol{p}_i\} = \{(x_i, y_i, z_i)\}. \tag{2.1}$$

As an example of scanning a surface, sampling points obtained with a laser scanning for a terra-cotta Shiisa are shown in Fig. 2.1.

In many cases a sampling point may have additional attributes such as a normal, intensity, color, and confidence of scanning. Among them, normal is often equipped (or easily obtained with a simple calculation as introduced in [HDD+92]) and plays important roles in many applications processing sampling points. Ideally speaking, normal equipped with a sampling point should be coincident with a normal at the corresponding point on the surface of the object. Confidence is a value indicating the accuracy of the scanning at each scanned point [CL96]. For instance, in a case scanning with a laser scanner, accuracy of points near highly curved regions or high specular regions tend to be low.

If the inner structure of an object is needed, required structure can be obtained as a volumetric data with using of X-ray computed tomography (CT) scanners or MRI scanners. Each sampling point of volumetric data is composed of coordinates and values like intensities. Volumetric data are, for instance, used for analyses of objects with multi material and medical engineering. Fig. 2.2 shows three slices of a CT-scanned data of an industrial object. Points with high intensities are indicated by white points while those with low intensities black.

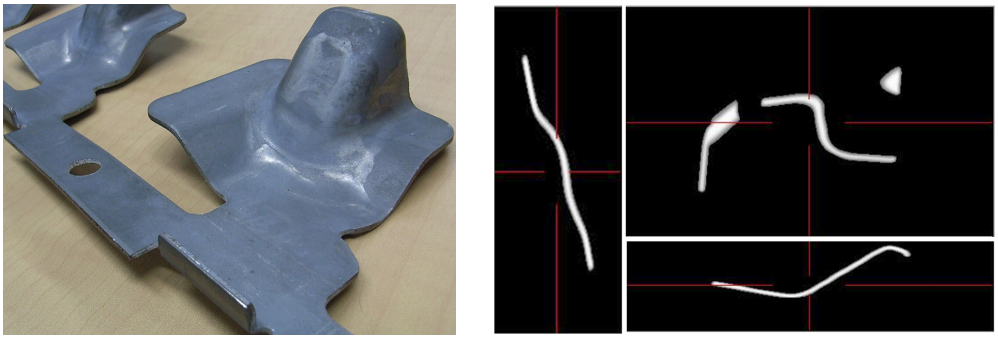


Fig. 2.2. Real object and three slices of CT-scanned data.

Unfortunately any type of scanned data generally includes some problematic parts. Noise, outliers, lack of sampling, and differences in density of sampling are typical problematic parts. Fig 2.3 shows examples of these problems. (a) shows **noise**, small oscillation, observed in an ear. **Outlier**, (b), is an extraordinary point far from surface. It is

caused by specular reflection of laser beam for instance. Area where laser beam cannot reach exists because of obstacles or shape of target object, and thus **lack of sampling** areas arise in scanned data. In Fig 2.3, this lack is observed in the mouth, (c). Scanned data of an object is generally obtained with aligning and merging several scanned data from distinct directions. Because of this iterative scanning, sampling density of an area which is scanned in several times may be dense than that of areas with single scanning. Thus scanned data has **differences in density of sampling**. The image in Fig 2.3 (d) shows the foot with large differences in sampling density. The quality of scanned data is determined according to the amount of such problematic parts. For instance, data with little amount of noise is high quality, in contrast data with many outliers, great amount of noise, many lack of sampling is considered as low quality.

In some applications, low quality scanned data causes many serious problems. One example of such a failure caused in generating a shape model directly from scanned data is displayed in Fig. 2.4. As the close-ups in the bottom row showing, some extra surfaces appear in the back of the head, the shape of the mouth is largely deformed because of the absence of sampling points, and the right forefoot cannot be reconstructed correctly as a result of different sampling densities.

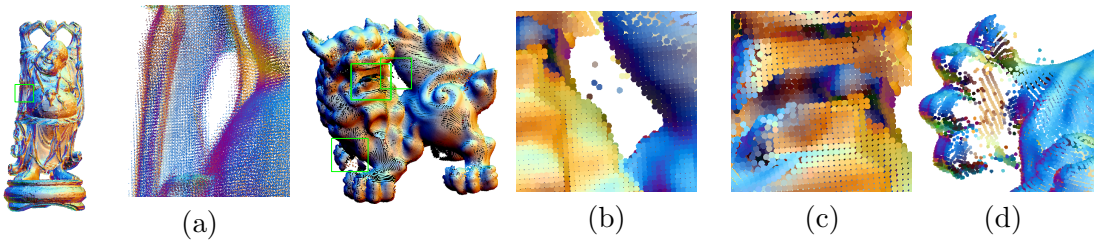


Fig. 2.3. Problems in scanned data.
(a) Noise. (b) Outliers. (c) Lack of sampling. (d) Differences in sampling density.

2.1.2 Shape Model

After scanning, measured data is converted to other shape representations which are more common ones in each application. In order to handle object shapes with computer effectively, many representations have been proposed and used. One of the most common representation is mesh, which is explained in the following.

Mesh

Because of its high ability of shape representation, mesh is adopted as a general representation method in areas involving science, engineering and industry. More specifically,

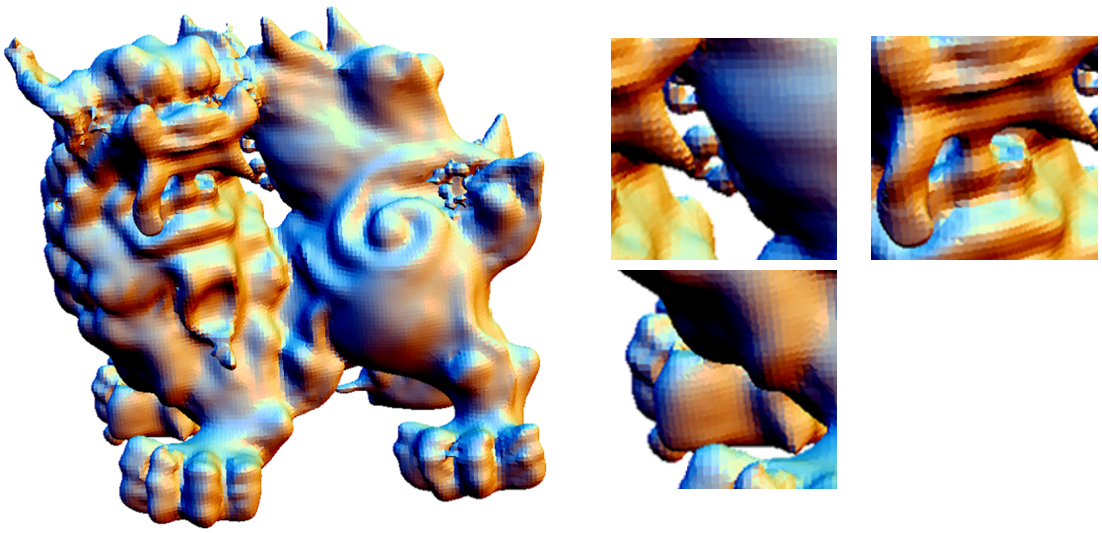


Fig. 2.4. Problems in surface reconstruction caused by low quality of scanned data.

computer graphics, computer aided design (CAD), and computer aided engineering (CAE) are instances. Formally, **mesh** M is a shape representation based on geometrical feature V and topological feature T:

$$M = (V, T). \tag{2.2}$$

Specifically, V is realized as coordinates of points and T means the connectivity of points. In terms of mesh, a point is called **vertex**. Connectivity is composed of connection between two vertices which defines an **edge**, and connection between more than three vertices which defines a **face**.

One of most common mesh is **surface mesh** which is a 2-manifold mesh in 3D space. Surface mesh whose all faces are triangle is referred to a **triangular mesh**. An example of a triangular mesh is in Fig. 2.5.

Another major type of mesh is **volume mesh** which fills a 3D region with a set of polyhedrons. Volume mesh whose all of the elements are tetrahedra is referred as a **tetrahedral mesh**, see an example in Fig. 2.6.

As shape representations mainly we treat triangular mesh and tetrahedral mesh in addition to spherically supported functions explained in the following chapter. In this thesis these two kinds of mesh work as destination or intermediate representations.

2.2 Related Work

At present, many technologies for processing shapes of objects are developed in large region including scientific and engineering. There are lots of interesting research areas performing geometry processing, and among them, we consider differential operators on

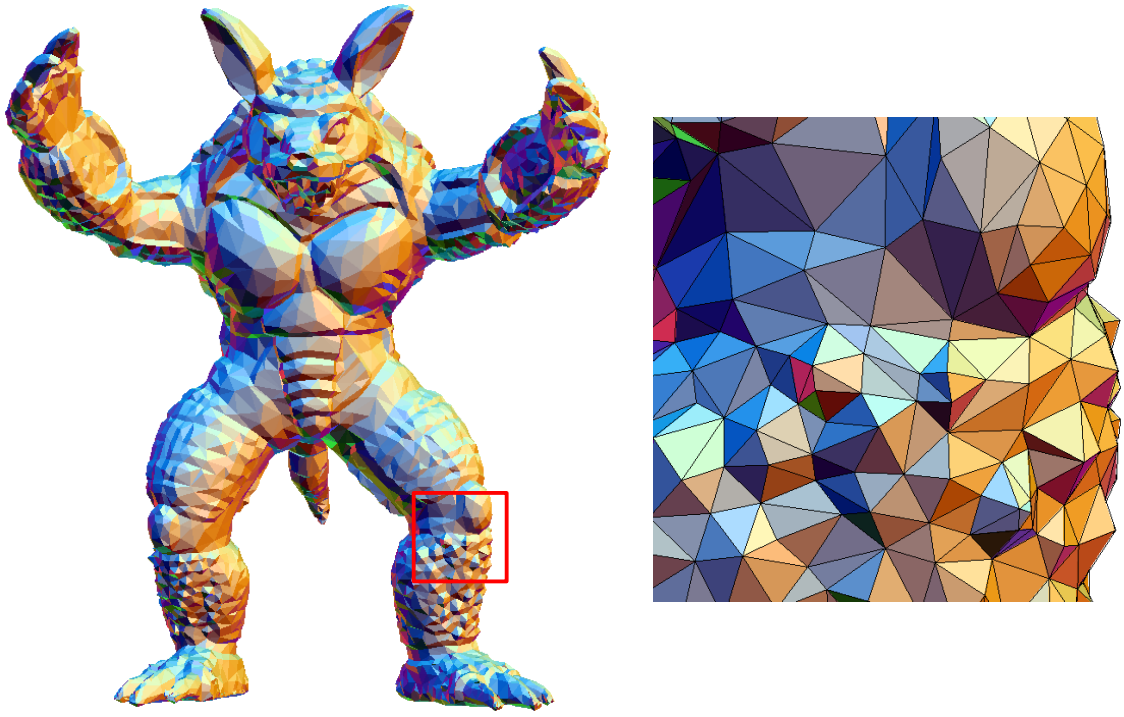


Fig. 2.5. Triangular mesh and its close-up of the boxed area.

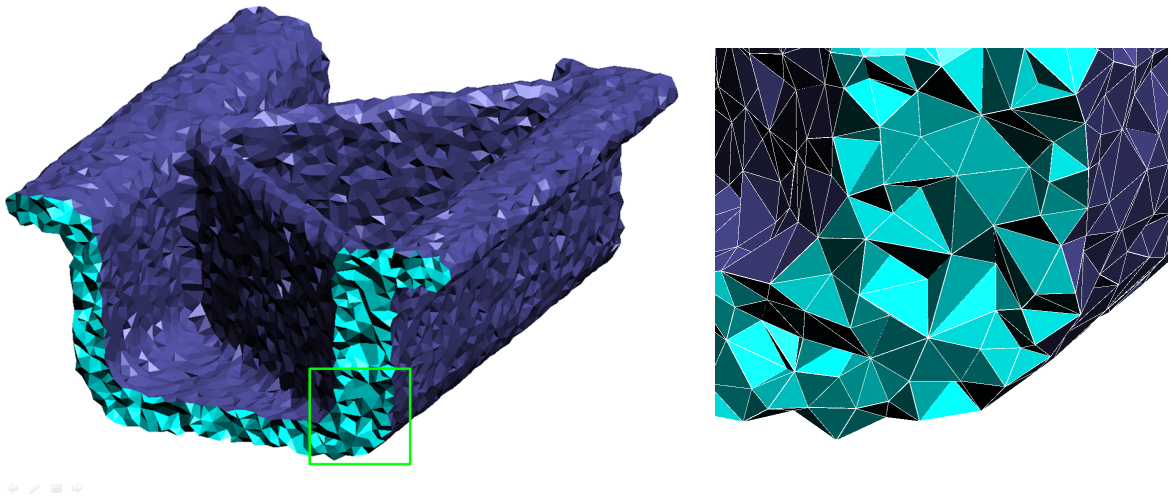


Fig. 2.6. Cross-section of a tetrahedral mesh and its close-up of the boxed area.

specific data structures, surface reconstruction, smoothing, and skeletal structure extraction. These areas have been well researched so far, and thus in all of these areas too many algorithms have been proposed for us to introduce all of them. So we focus only on previous studies which are major or have close relations to our research.

2.2.1 Differential Operators

The continuous differential operators, gradient, divergence, and Laplacian operators, are for a orthogonal grid. Recently, differential operators for a triangular mesh and tetrahedral mesh have been proposed in [DMSB99] and [TLHD03] respectively. Since we are working on a set of spherical supported functions, using differential operators specialized for this structure is considered as a better strategy, however, no differential operators for a set of spherical supported functions have been proposed yet.

2.2.2 Surface Reconstruction

For the past several decades, many algorithms have been proposed because of the various demands of the relevant areas of application. Previous algorithms can be roughly classified into two groups according to their approach: explicit methods and implicit methods.

Explicit Methods

Typical methods in this category involve Delaunay-based algorithms. Basically, Delaunay-based algorithms generate meshes whose vertices are parts of sampling points. This is an intuitive approach, and the first algorithm for surface reconstruction proposed by Boissonnat [Boi84] is also categorized into this group. Delaunay-based algorithms, as shown in its name, use Delaunay triangulations of sampling points or geometric structure related to those. Due to the nature of these structures, some algorithms can generate guaranteed mesh in terms of approximation accuracies, topological characteristics, or qualities like aspect ratio. Unfortunately, Delaunay triangulations are very sensitive to noise existing in most scanned data. Recently, a number of algorithms with noise robustness have developed: the Power Crust [ACK01] developed by Amenta et al., the Robust Cocone [DG06] proposed by Dey and Goswami, and the Eigencrust algorithm [KSO04] constructed by Kolluri et al. Although these are more robust than previous ones, implicit methods provide much better results with respect to noise robustness.

Moreover, Delaunay-based algorithms assume that sampling points are sufficiently dense. Their demands to the sampling density is related to the curvature of objects; highly curved region is needed to be densely sampled. In areas with extremely high curvatures such as sharp edges and corners, infinitely dense sampling is required theoretically.

In addition, algorithms in this category tend to be time-intensive because of the global nature of Delaunay triangulations.

Implicit Methods

Nowadays, implicit approaches are a major force in surface reconstruction because of their

stability with noise and low computational costs. Algorithm in this category starts from the method developed by Hoppe et al. [HDD+92].

Employing implicit methods, the space is divided into the inside and outside of the object. Thus a surface is defined as the boundary between the internal and external points. Surface is offered just as an implicit form, and therefore, if necessary, it is approximated with a mesh generated with a polygonizer such as Marching cubes [LC87], dual contouring [JLSW02], and other polygonization method [Blo88]. Implicit algorithms are categorized more precisely with respect to their local or global natures.

Local implicit methods. Because of their locality, local implicit methods have advantages such as highly accurate representation and low computational cost, and thus local implicit methods can process large data sets with several giga bytes. Algorithms in this group began with VRIP [CL96] which can provides high-resolution surfaces but struggles to handle misaligned surfaces. Other instances of local implicit approaches are algorithms employing moving least squares [ABCO+01, GG07] and its variants [AK04, ÖGG09]. Multi-level Partition of Unity implicits (MPU) [OBA+03] proposed by Ohtake et al. excel in the above advantages.

Local implicit methods have many merits including those listed above and even noise robustness owing to the implicit nature. But because of their locality, there are some difficulties in handling very low-quality data such as that with large amounts of noise, outliers, lack of sampling, and large differences in sampling density. The resulting shapes for low quality data are sometimes significantly deformed, and may have some extra components. The fails shown in Fig. 2.4 are results obtained with a local implicit method.

Global implicit methods. Global implicit approaches have the advantage of handling low quality data. Examples of such methods include the Radial Basis Function (RBF) approach [CBC+01] developed by Carr et al., the integration of Voronoi diagrams and the variational method [ACSTD07] introduced by Alliez et al., the Graph-cut approach given by Hornung and Kobbelt [HK06], and the finite element method proposed by Sharf et al. [SLS+07]. The Poisson surface reconstruction technique proposed by Kazhdan et al. [KBH06] especially excels in computational speed and noise robustness. Unfortunately, the global implicit approach achieves noise robustness at the cost of high-accuracy reconstruction. Fig. 2.7, the results with a global implicit method for the sampling points in Fig. 2.1 show these characteristics well. Parts like the point of beard and the mouth which correspond to the lacks of sampling are largely deformed. In addition, because of the global nature, their smoothing effect influences other regions. As a result the patterns on the toe are smoothed out.

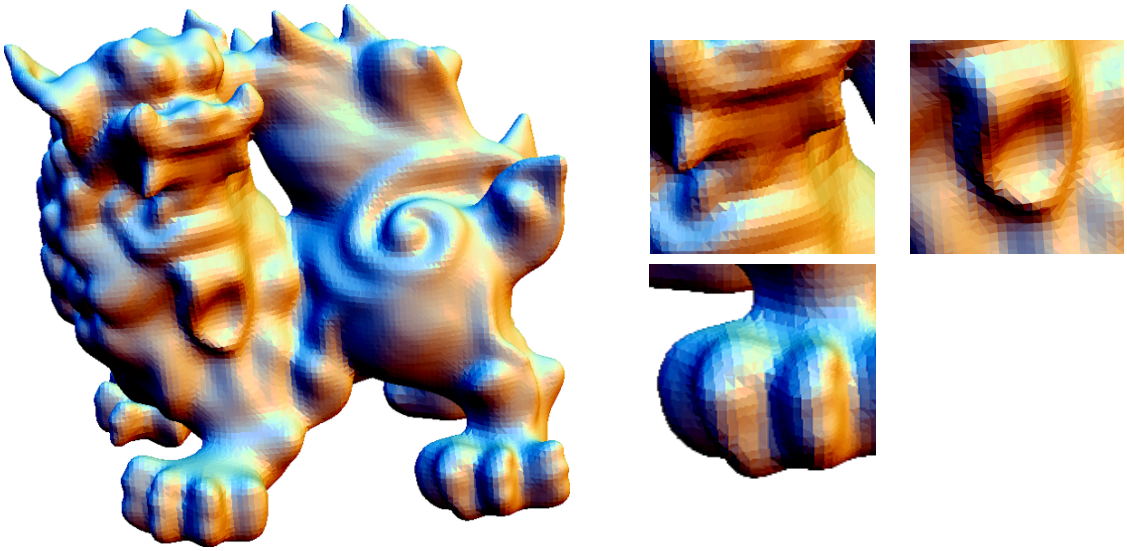


Fig. 2.7. Reconstructed model with a global implicit method.

2.2.3 Smoothing

The above discussion about surface reconstruction suggests that generating smooth and precise mesh through use of only surface reconstruction is a hard issue. One solution to improve the smoothness is smoothing which has been well-studied as surface reconstruction. So many smoothing algorithms have been proposed, for both of surface mesh and volumetric grid.

The simple and intuitive strategy is moving vertices in the direction so that geometrical differences with surrounding vertices become small. As an instance of this method, Laplacian smoothing is the most intuitive and simple one, so it is followed by many extensions such as Taubin Smoothing [Tau95], bi-Laplacian smoothing [KCVS98] constructed by Kobbelt et al., and mean-curvature flow [DMSB99] introduced by Desbrun et al. Fig. 2.8 shows the effect of bi-Laplacian smoothing. The left image shows the initial surface mesh which many undesired bumps are observed on. This smoothing makes the surface more smooth with only moving mesh vertices as shown in the right image.

More sophisticated techniques are normal-based smoothing [Tau01, OBS02, CT03]. These algorithms smooth normal vectors first. Then these move vertices. A similar method for grids is proposed by Tasdizen et al. [TWBO02] in the basis of level set methods [Set99].

An disadvantage of mesh smoothing is that they cannot change the topology of target mesh, and thus extra components like ones in Fig. 2.4 cannot be removed.

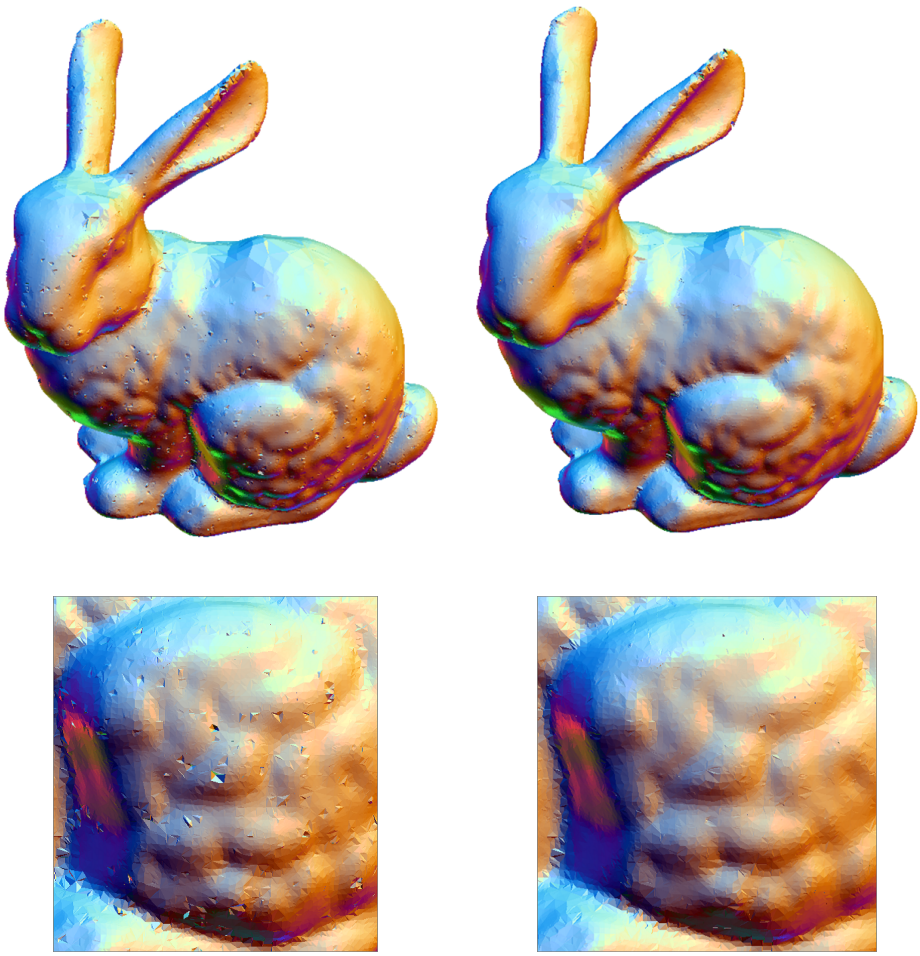


Fig. 2.8. Results of surface mesh smoothing.
Left: initial surface mesh and close-up of the left leg.
Right: results of smoothing for the mesh in the left image.

2.2.4 Skeletal Structure

There are various kinds of skeletal structures for many application areas. As reviewed in [CSM07], many algorithms have been developed for extracting skeletal structures. These are briefly reviewed below.

To obtain voxelized 1D/2D skeletons from a voxelized solid, method [JBC07] proposed by Ju et al. performs a thinning algorithm. Another approach [PH02] proposed by Prohaska and Hege applies a distance transformation from the boundary voxels. Then the voxels taking the local maxima of the distance field are detected. These techniques extract skeletal structures as voxels. Thus an additional polygonal mesh generation method as [FSKK05] is required if the output model is desired as a mesh representation.

A polygonal approximation of the medial axis can be directly obtained using a Voronoi

diagram of sampling points on a solid boundary [ACK01, DZ03]. Although these techniques do not require grid sampling inside the solid, the topology of the resulting mesh is sensitive to small perturbations of the solid boundary. These methods are therefore not suited to processing scanned objects, which typically have noisy boundaries.

Using a potential field [MWO03, CSYB05] is another possible approach. The skeletal points are extracted as the stationary points of a gradient descent flow defined by the sum of radial functions centered at points on a given solid boundary. This approach is mainly used for extracting 1D skeletal curves.

Chapter 3

Function on Spherical Covering

3.1 Introduction

Given a set of points $\{\boldsymbol{p}_j \in \mathbb{R}^n\}$ which may be equipped with some values, a scalar field for a domain in $\mathbb{R}^3$ including $\{\boldsymbol{p}_j\}$ can be generated. The instance of such a set of points are, for example, CT-scanned data and surface scanned data, and values attached to each point are, intensity and normal vector respectively. An example of a generated scalar field for 2D points is shown in Fig. 3.1. Colors indicates a high value with red and low with blue. Such a scalar field can be generated with covering a domain with spheres equipped with local functions.

In this chapter, the structures of such a sphere and a cover is described in Section 3.2 and the properties of the scalar field are explained in Section 3.3. The generation algorithm for 2D and 3D sampling points are denoted in Section 3.4.1 and 3.4.2 respectively. Based on a spherical cover, dividing a covered domain with tetrahedral mesh is also possible. In Section 3.5, a tetrahedral meshing is explained.

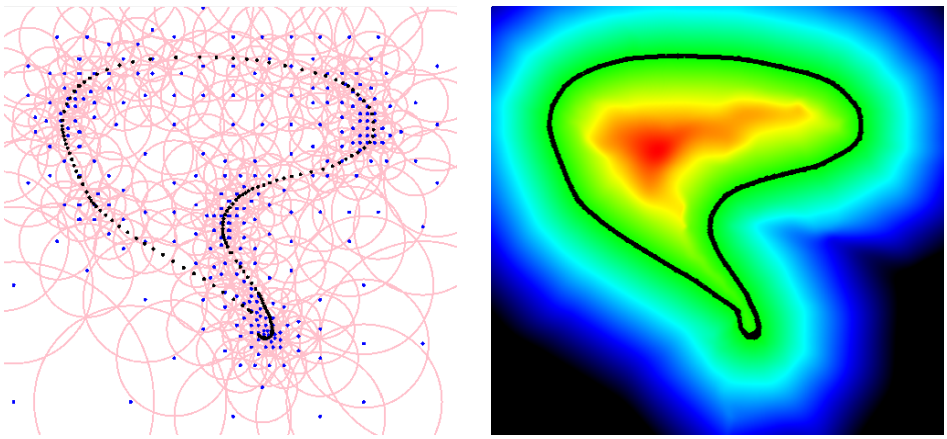


Fig. 3.1. Shape representation applying spherical covering method in 2D.
Left: a set of sampling points and a spherical cover.
Right: generated scalar field and isosurface representing the shape of the original object.

3.2 Data Structure

Spherical support

Below the coordinates of a point x are indicated with a bold letter $\boldsymbol{x}$. **Spherical support** s_i is an n-disk which is defined by center $\boldsymbol{c}_i$ and radius r_i, and has an implicit function $g_i(\boldsymbol{x})$. The **n-disk** (also called as **n-ball**) is the interior of a **sphere** which is the set of all points whose distance from $\boldsymbol{c}_i$ is r_i. n is the dimension of a point in $\{\boldsymbol{p}_i\}$. **Support** of a function $f(x)$ is a closure of the set of argument x which satisfies $f(x) \neq 0$. The support of $g_i(\boldsymbol{x})$ should be s_i itself. Implicit function $g_i(\boldsymbol{x})$ locally approximate the values associated points inside s_i such as intensity or distance from a surface of an original real object.

In this thesis, a simpler term **support** means spherical support unless stated otherwise.

Spherical Cover

When a union of spherical supports contains a domain, the set of spherical supports is referred to as a **spherical cover** of the domain, and the domain is **covered** by the spherical supports. Note that a covered domain is completely contained in the union of supports. Fig 3.2 shows examples of spherical covers for 2D and 3D domains including sets of points.

In order to avoid gaps in a spherical cover (areas which are not covered with spheres) as long as possible, spherical supports should intersect each other. Moreover, with considering these intersections, we can define the connectivity between the spherical supports. When a support intersects with another support, these two supports are considered as **connected**. Moreover we call a spherical support intersecting with s_i as **neighbor** of s_i. Fig. 3.3 shows examples of this connectivity rules; support s_i and s_j is connected and support s_j and s_k is connected, but support s_i and s_k is connected. This connectivity is, in a manner of speaking, a semi-connectivity relation which means that connectivity between arbitrary two supports cannot be determined by transitive law. For example, in Fig. 3.3, connectivity between support s_i and s_k can not obtained with the connectivity of s_i and s_j, and s_j and s_k.

For connected supports s_i and s_j, we define **intersection disk** as a disk whose boundary is the circle where ∂s_i and ∂s_j intersect. See Fig. 3.3. Intersection disk of s_i and s_j is shown in a green line.

The implementation details of connectivity of support spheres are explained in Appendix B. It helps fast and memory efficient computation.

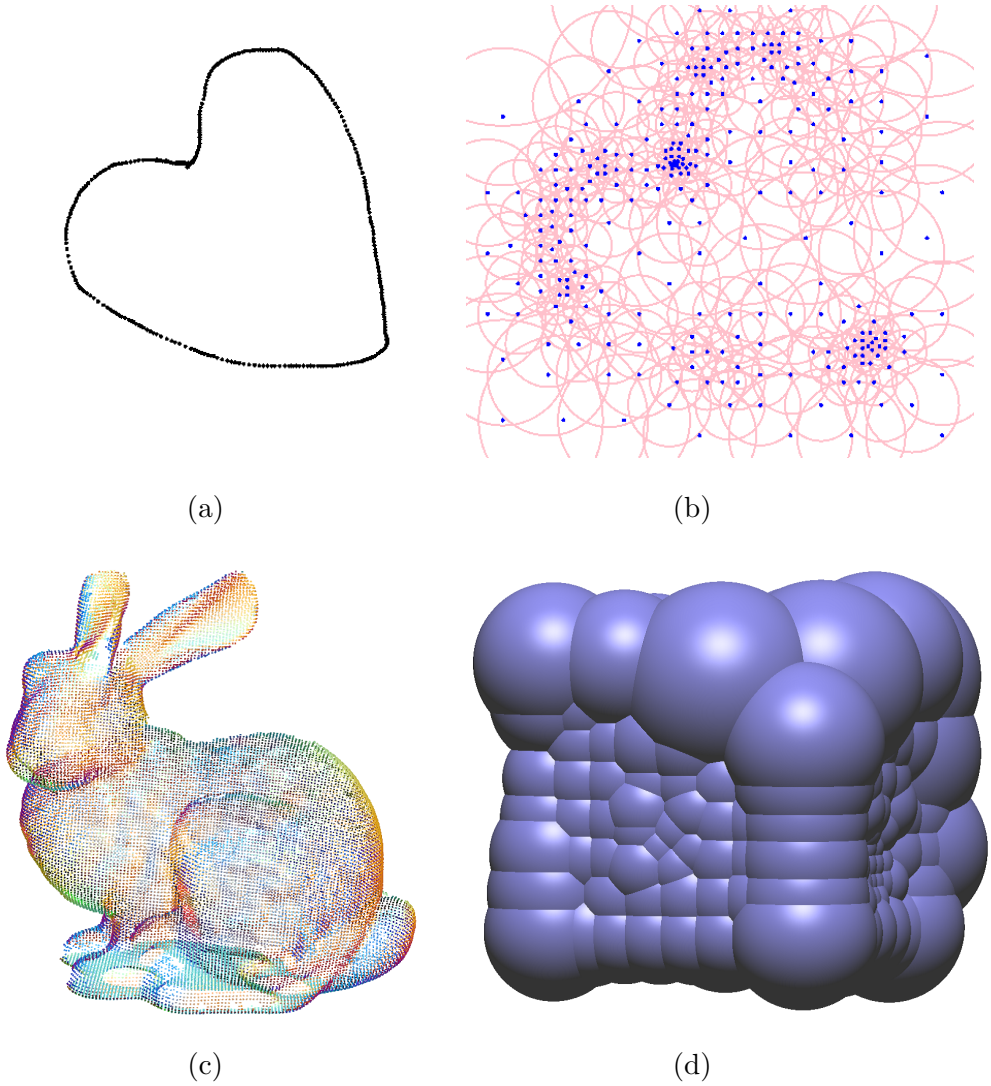


Fig. 3.2. Examples of spherical covers for 2D ((a) and (b)) and 3D ((c) and (d)) domains.

3.3 Partition of Unity

Given a spherical cover $\{s_i\}$, an weighted average $f(\boldsymbol{x})$ of local approximation functions $\{g_i(\boldsymbol{x})\}$ can be obtained. $f(\boldsymbol{x})$ is described as follows:

$$f(\boldsymbol{x}) = \frac{\sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x})}{\sum_i w_i(\boldsymbol{x})} \tag{3.1}$$

where $w_i(\boldsymbol{x})$ is a weighting function whose support is s_i similar to $g_i(\boldsymbol{x})$. The sum on the denominator is for normalization of the weight. The domain of $f(\boldsymbol{x})$ is a union of $\{s_i\}$.

This is the algorithm known as **Partition of Unity** [Ren88], which is hereafter abbreviated as **PU**. This name comes from the normalization term.

The approximation error of $g_i(\boldsymbol{x})$ is bounded up to a given tolerance ε. Because of the nature of PU, the error of $f(\boldsymbol{x})$ is also bounded up to ε.

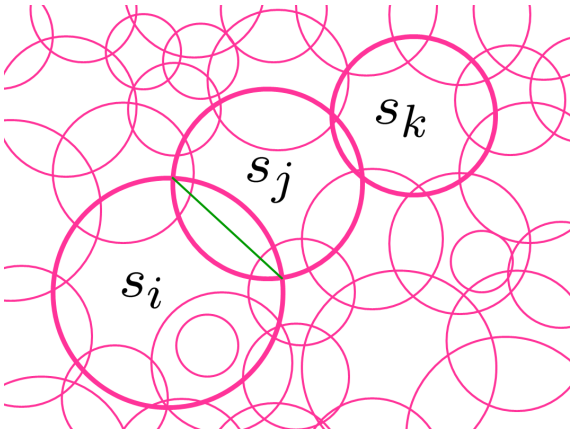


Fig. 3.3. Connectivity of spherical supports. Intersection disk of s_i and s_j is shown in green line.

3.4 Generation of Spherical Cover

In this section, algorithms to generate spherical covers for sampling points from surfaces and for volumetric data are explained respectively. Section 3.4.1 is for sampling points from surfaces and Section 3.4.2 is for volumetric data.

3.4.1 Spherical Cover for Sampling Points from Surface

The aim of generating a spherical cover from sampling points from a surface is representing the shape of an object. First, the representation based on spherical covering is denoted. Next, the details of generation methods of a support sphere and a whole spherical cover are explained.

Shape Representation with Spherical Cover

Given a set of points $\{\boldsymbol{p}_j\}$ sampled from a surface of an object associated with oriented normals $\{\boldsymbol{n}_j\}$, we approximate the surface of the object with the zero-level surface of signed distance function $f(\boldsymbol{x})$. In our algorithm, points inside the object have positive values of $f(\boldsymbol{x})$. An example of a scalar field of $f(\boldsymbol{x})$ is shown in the image on the left of Fig. 3.4. Values of $f(\boldsymbol{x})$ are indicated in red for high, and blue for low. A domain including all the sampling points is covered by a spherical cover as shown in the image on the right of Fig. 3.4. This approach is similar to that of sparse low-degree polynomial implicit (SLIM) surfaces [OBA05] but SLIM generates a spherical cover only for the boundary surface. On the other hand, in this algorithm, spherical supports are generated inside and outside as well as on the boundary surface.

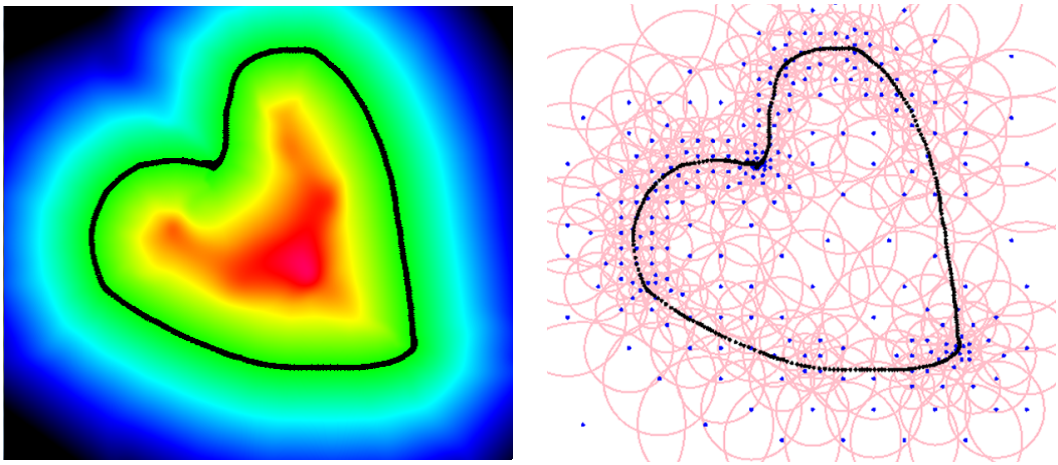


Fig. 3.4. Examples of a scalar field and a spherical cover in 2D.
Left: the scalar field determined by $f(\boldsymbol{x})$ and the reconstructed image (black line).
Right: the supports of local approximation functions $\{g_i(\boldsymbol{x})\}$ are indicated with pink circles whose centers are represented by blue dots. Black dots means the sampling points.

Spherical Support

As mentioned above, a spherical support s_i is defined with the center $\boldsymbol{c}_i$ and the radius r_i. Support s_i has a local implicit function $g_i(\boldsymbol{x})$, whose support is s_i itself. The local function $g_i(\boldsymbol{x})$ approximates the signed distance from the surface inside s_i, see the left image of Fig. 3.5. We adopt a linear function as $g_i(\boldsymbol{x})$. The reasons of this are that, mesh, the final representation of our algorithm, is a linear approximation of a shape, and its density is directly determined by the density of supports. Since we generate a spherical cover with respect to the approximation error, adapting linear functions as approximation functions is reasonable. Local approximation function $g_i(\boldsymbol{x})$ is represented as $g_i(\boldsymbol{x}) = \boldsymbol{\alpha}_i \cdot (\boldsymbol{x} - \boldsymbol{c}_i) + \beta_i$ with coefficient vector $\boldsymbol{\alpha}_i$ and constant term β_i. The value of $g_i(\boldsymbol{x})$ for a point $\boldsymbol{x}$ inside s_i locally approximates the signed distance of $\boldsymbol{x}$ from the surface. Isosurface $g_i(\boldsymbol{x}) = 0$ therefore approximates the surface inside s_i.

Local approximation $g_i(\boldsymbol{x})$ can be determined with least square fitting for sampling points $\{\boldsymbol{p}_j\}$ inside s_i. The obtained $\boldsymbol{\alpha}_i$ and β_i are described as

$$\boldsymbol{\alpha}_i = \frac{\sum_j w_i(\boldsymbol{p}_j)\boldsymbol{n}_j}{\|\sum_j w_i(\boldsymbol{p}_j)\boldsymbol{n}_j\|}, \quad \beta_i = \boldsymbol{\alpha}_i \cdot \left(\frac{\sum_j w_i(\boldsymbol{p}_j)(\boldsymbol{c}_i - \boldsymbol{p}_j)}{\sum_j w_i(\boldsymbol{p}_j)} \right). \tag{3.2}$$

These are the weighted averages of normals and relative coordinates for $\boldsymbol{c}_i$ of $\{\boldsymbol{p}_j\}$. In this algorithm, weighting function $w_i(\boldsymbol{x})$ is the second-degree B-spline $b_2(d)$:

$$w_i(\boldsymbol{x}) = b_2\left(\frac{3\|\boldsymbol{x} - \boldsymbol{c}_i\|}{2r_i} \right), \tag{3.3}$$

$$b_2(d) = \begin{cases} 0 & \left(\frac{3}{2} < d\right) \\ \frac{1}{2}\left(\frac{3}{2} - d\right)^2 & \left(\frac{1}{2} < d \leq \frac{3}{2}\right) \\ -d^2 + \frac{3}{4} & (otherwise) \end{cases} . \tag{3.4}$$

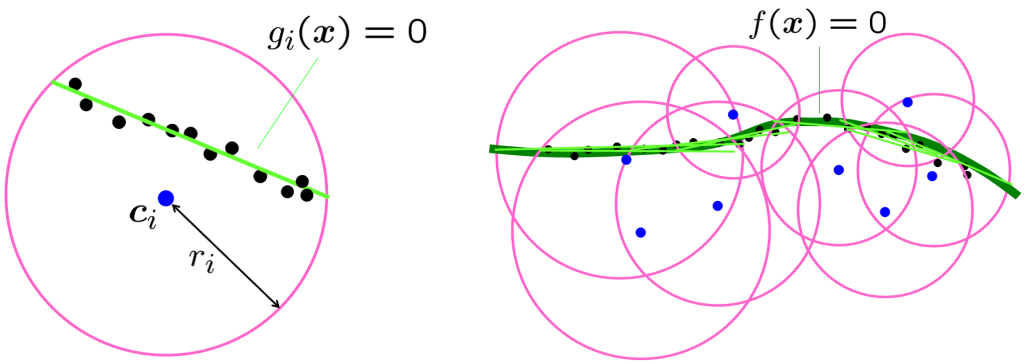


Fig. 3.5. A spherical support and a spherical cover for surface reconstruction. Black points mean sampling points.
Left: a spherical support with a local approximation function $g_i(\boldsymbol{x})$ shown in a yellow green line.
Right: a spherical cover. The weighted average $f(\boldsymbol{x})$ of $\{g_i(\boldsymbol{x})\}$ is show with a green line.

Spherical Cover

On a spherical cover, a weighted average $f(\boldsymbol{x})$ of local approximations $\{g_i(\boldsymbol{x})\}$ can be determined. It represents the signed distance from arbitrary point $\boldsymbol{x}$ (inside the spherically covered domain) to the surface of the object. Here is the equation:

$$f(\boldsymbol{x}) = \frac{\sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x})}{\sum_i w_i(\boldsymbol{x})}. \tag{3.5}$$

The isosurface $f(\boldsymbol{x}) = 0$ approximates the shape of an object as the right image of Fig 3.5.

Here we outline the generation of a spherical cover for a domain including sampling points $\{\boldsymbol{p}_j\}$. We assume that the domain is a rescaled bounding box of $\{\boldsymbol{p}_j\}$ the length of whose longest edge equals to one. In this step, a spherical cover is generated with two-step algorithms. We realize the initial step using the method proposed by MPU [OBA+03] first, and then modify it employing a error-driven procedure. The number of final spherical supports becomes lower than the result generated through MPU.

Similarly to MPU, the generation process starts with dividing the domain with an adaptive octree measuring local approximation errors, details below. Using the centers of cells (octants) of an adaptive octree as the centers of supports is a good way of ensuring the accuracy of reconstruction. The difference of the division level with a neighboring cell

is set to at most one. The centers of quadtree (2D version of a octree) for 2D sampling points shown in the left image of Fig. 3.6 is shown in the center image. The centers exist following the sampling density. The finally obtained spherical cover is shown in the right image. The centers of supports are shown with blue points. The distribution of supports and those size reflect the density of sampling points.

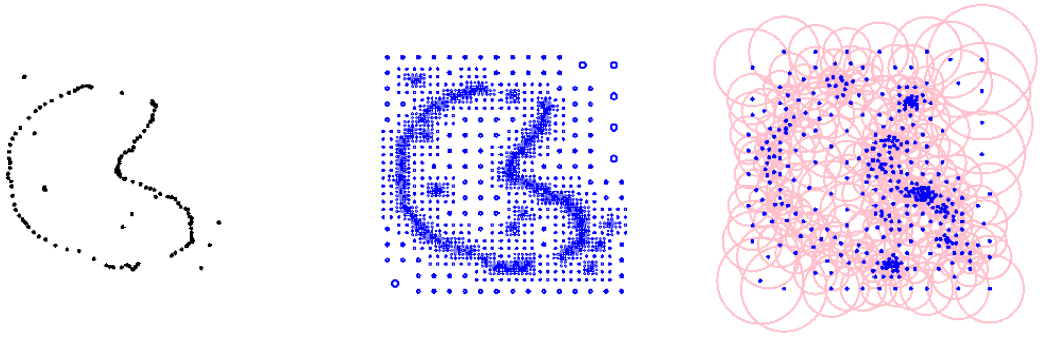


Fig. 3.6. Centers of quadtree and generated spherical supports in 2D.
Left: input sampling points.
Center: the centers of cells of a quadtree.
Right: generated spherical supports and those centers.

In the method to generate the initial spherical support, an octant is assigned a spherical support s_i whose center $\boldsymbol{c}_i$ is the center of the octant and radius r_i is 0.75 times the length of the octant's diagonal. Then, $g_i(\boldsymbol{x})$ is determined with equation (3.2). We measure the approximation error of $g_i(\boldsymbol{p}_j)$ for sampling points $\{\boldsymbol{p}_j\}$ inside the corresponding support with the maximum value of $|g_i(\boldsymbol{p}_j)|$. Each octant is divided recursively until its approximation error does not exceed a user-specified tolerance ε.

After the initial spherical cover is generated, it is modified with expansion of the radii of randomly selected supports as long as those local approximation errors do not exceed ε. Spherical supports completely included inside an expanded support are eliminated. In Fig. 3.7, 2D examples show this expansion. The left image shows the initial spherical cover. The red support is to be expanded. After the expansion, as shown in the right image, supports inside the red support (shown in purple) are eliminated. This operation means maximizing r_i without exceeding ε. This expansion and elimination step helps to reduce the number of supports. In our experiments, the final number of supports is about half the number of octants.

3.4.2 Spherical Cover for Volumetric Data

In this section, a set of spherically supported functions are constructed from volumetric data. The goal of this covering is approximation of the values of a subset of input points.

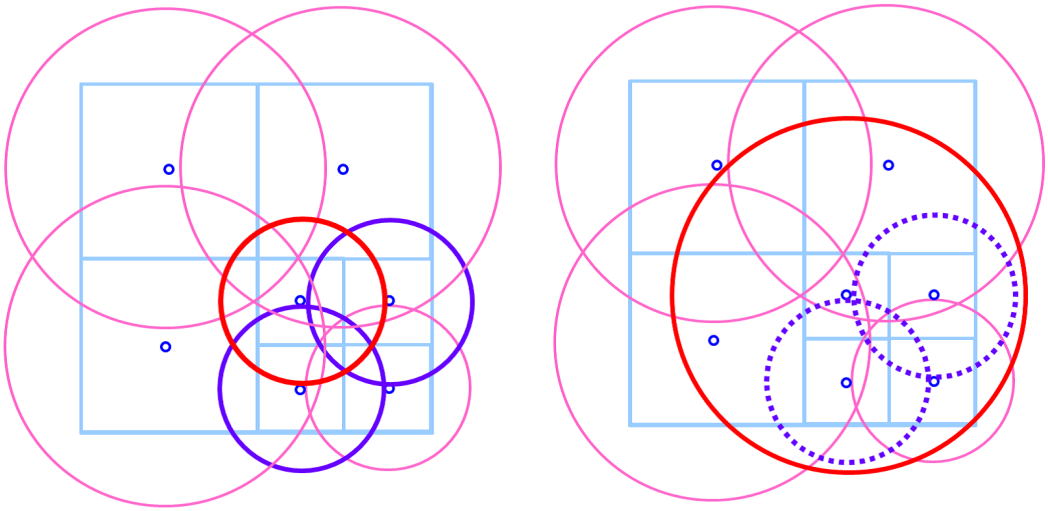


Fig. 3.7. Spherical support expansion and elimination.
Left: the initial spherical cover generated by MPU.
Right: the spherical cover after expansion.

Although we explain only the case that covering an object, it is possible to cover the whole volumetric data in the same manner.

Input

This algorithm takes a set of spatial points equipped with normalized values as input data:

$$\mathscr{P} = \{\boldsymbol{p}_i = (\boldsymbol{x}_i, v_i) | \boldsymbol{x}_i \in \mathbb{R}^3, v_i \in [0,1]\} \tag{3.6}$$

These points can be obtained with normalization of intensities of sampling image including CT scanned data and grayscale image. The left image of Fig. 3.8 shows a 2D analogous example. Such a set consists of two kinds of points; points belonging to an object and points that make up the background. We assume that points with a value greater than a user's specification T represent an object. Taking T as input is reasonable because each material has a unique CT value. In the following of this chapter, these points are referred to as object points, and the set of all object points is described as

$$\mathscr{P}_{\mathrm{obj}} = \{\boldsymbol{p}_i | v_i > T\}. \tag{3.7}$$

With setting the value of $\mathscr{P}_{\mathrm{obj}}$ to 0, a cover for the whole volumetric data can be obtained. This algorithm construct a set of spherical supports covering $\{\mathscr{P}_{\mathrm{obj}}\}$ and its surrounding space.

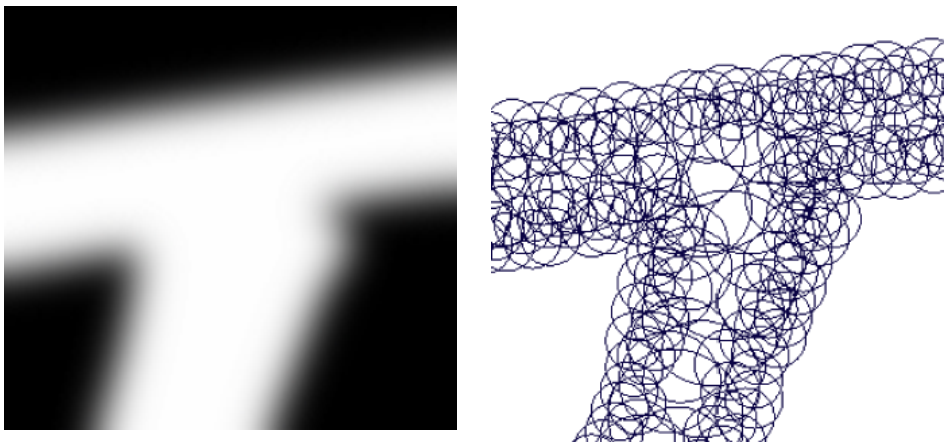


Fig. 3.8. Spherical cover for volumetric data.
Left: grayscale image (volumetric data).
Right: Support circles.

Spherical Support

As a local approximation function $g_i(\boldsymbol{x})$ associated to each spherical support s_i, we adopt a quadratic function which is described as

$$\begin{array}{c} g_i(\boldsymbol{x}) = c_{xx}x^2 + c_{yy}y^2 + c_{zz}z^2 + c_{xy}xy + c_{yz}yz \\ + c_{zx}zx + c_x x + c_y y + c_z z + c_0 \end{array}, \tag{3.8}$$

where each coefficient c_* is in $\mathbb{R}$. The center $\boldsymbol{c}_i$ of s_i is a point among $\mathscr{P}_{\mathrm{obj}}$. The radius r_i is adapted to the change of the intensities in the vicinity of $\boldsymbol{c}_i$ as in the right image of Fig. 3.8. The values of intensities rapidly change near boundary, as a result the sizes of supports also largely change.

Spherical Cover

In order to approximate the intensities of a part of $\mathscr{P}$ with a scalar function $f(\boldsymbol{x})$, a set of spherically and adaptively supported functions $\{g_i(\boldsymbol{x})\}$ is generated. See Fig. 3.9, an image of the relation of an approximation function $f(\boldsymbol{x})$ and a set of local approximations $\{g(\boldsymbol{x})\}$.

The approximation function for a point $\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3$ is represented with a partition of unity using polynomial approximations [Ren88]:

$$f(\boldsymbol{x}) = \frac{\sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x})}{\sum_i w_i(\boldsymbol{x})}. \tag{3.9}$$

Weight function $w_i(\boldsymbol{x})$ is specified for each spherical support. For a support s_i with a radius r_i and a center $\boldsymbol{c}_i$, $w_i(\boldsymbol{x})$ is defined as

$$w_{r_i}(\boldsymbol{x}) = \begin{cases} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{c}_i\|^2}{2r_i^2}\right) \Big/ (2\pi)^{\frac{2}{3}} r_i^2 & (\|\boldsymbol{x}-\boldsymbol{c}_i\| \le r_i) \\ 0 & (otherwise) \end{cases}. \tag{3.10}$$

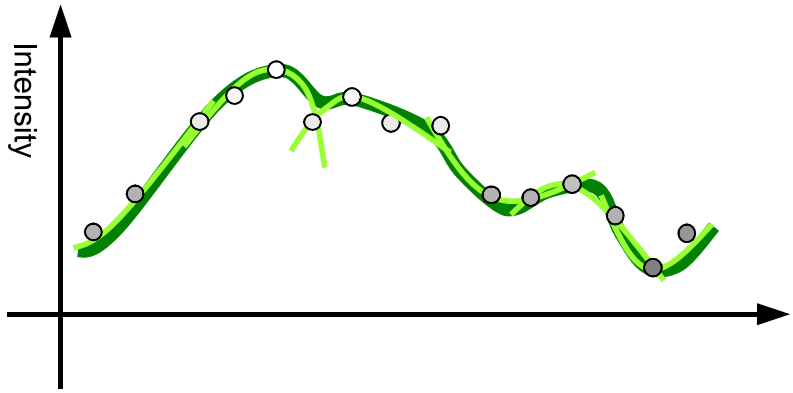


Fig. 3.9. Approximation of intensity with second degree local functions. Green line means approximation function $f(\boldsymbol{x})$ and yellow green lines mean local approximation functions $\{g_i(\boldsymbol{x})\}$.

In equation (3.10), the support of $w_i(\boldsymbol{x})$ is truncated in $\|\boldsymbol{x} - \boldsymbol{c}_i\| = r_i$. This truncation may cause a discontinuity of $f(\boldsymbol{x})$. In order to make $f(\boldsymbol{x})$ continuous, if needed, cubic B-Spline may be a good candidate $w_i(\boldsymbol{x})$. We experimented a B-Spline version but could not find discernible differences of the quality of results.

The details to decide the radius r_i and local function $g_i(\boldsymbol{x})$ of s_i is denoted in the following section. Then the whole algorithm to generate a spherical cover will be explained.

Generation of Spherical Support

Here we denote the way to generate single spherical support s_i. Local function $g_i(\boldsymbol{x})$ associated with s_i approximates intensity of points $\{\boldsymbol{p}_j\}$ inside s_i. Since original CT data include noise, using approximated data is better to achieve noise robustness. We generate a spherical support following the error-driven approach proposed in [OBS03]. Given a noise tolerance ε, the support radius r_i of s_i is determined as the solution of the following equation:

$$E_i(r_i) = \varepsilon \tag{3.11}$$

where

$$E_i(r)^2 \equiv \left. \frac{\sum_j w_j(\boldsymbol{x})(g_i(\boldsymbol{x}_j) - v_j)^2}{\sum_j w_j(\boldsymbol{x})} \right|_{g_i(\boldsymbol{x}) = \underset{g_i}{\operatorname{argmin}} E_i^2(r)} . \tag{3.12}$$

We sum over $\{\boldsymbol{p}_j\}$, the points whose distances to the center $\boldsymbol{c}_i$ of s_i are less than or equal to r. With respect to the definition (3.12), $g_i(\boldsymbol{x})$ is a minimizer of $E_i(r)$.

Usually, error tolerance ε is offered as a scanning information. But in the case that ε is not available, we set the value as 5% of the range of the values of intensity.

Equation (3.11) means to maximize the radius of a support without exceeding the error tolerance. Under the assumption that the approximation error monotonically increases as r grows larger, equation (3.11) can be solved with a bisection method [PTVF93]. Let us denote the minimum and maximum radii used in the bisection method as $r_{\min}$ and $r_{\max}$ respectively. Setting a large value for r_{max} enables reducing the number of supports. In our implementation, $r_{\max}$ is set to the maximum thickness of the object, in most cases which is directly obtained from the scanned object itself. If thickness cannot be known, a user can estimate it through a distance transformation for the inside of the object. On the other hand $r_{\min}$ should be small enough with keeping the set of supports covering the object without gaps.

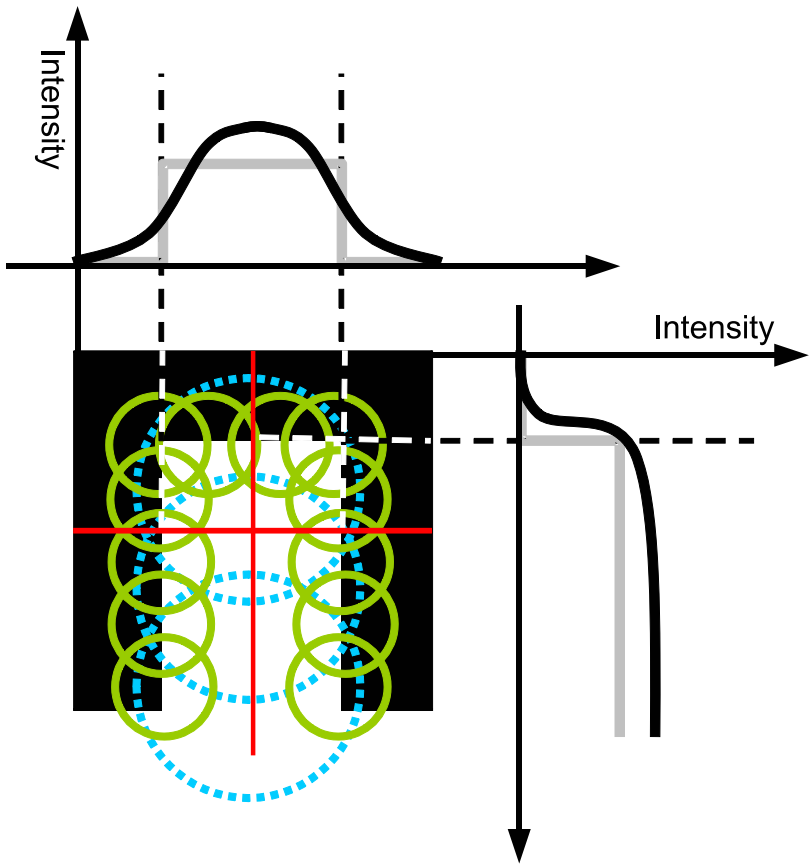


Fig. 3.10. Spherical supports and approximated intensities of CT-scanned data.
Lower left: a part of CT-scanned data and generated spherical supports.
Top and right: graphs of approximated intensities.

Because the intensity changes rapidly on boundaries of object, the radii of spherical supports whose centers are near a boundary tend to be small. In contrast, far enough from the boundary, the intensity changes slightly or may not change in supports, and

therefore the radii become large. Fig. 3.10 this tendency of change of the radii. A part of CT-scanned data is also displayed in lower left. The radius of a small green circle (2D spherical support) means the minimum radius $r_{\min}$, while the radius of a larger circle drawn in blue dotted line means the maximum radius $r_{\max}$. The top graph shows the changes in the intensity at the horizontal cross section, while the graph on the right shows that at the vertical. These two cross sections are drawn with red lines in the image of CT-scanned data. As shown in these graphs, the changes in the approximated intensity are similar to Gaussian functions.

Generation of Spherical Cover

In this section, a method to generate a spherical cover like that in Fig. 3.11 is explained. Generated spherical cover covers a space including $\mathscr{P}_{\text{obj}}$. The algorithm is summarized below.

Algorithm: generating spherical cover

1. Initialization: set $\mathscr{P}_{\text{obj}}$ as the center candidate list $\mathscr{C}$.
   Mark all points in $\mathscr{C}$ as *uncovered*
2. Spherical support generation: select a point randomly from $\mathscr{C}$ as a center $\boldsymbol{c}_i$
   and decide the radius r_i and local function $g_i(\boldsymbol{x})$
3. Covering check: remove covered points from $\mathscr{C}$
4. Termination check: if $\mathscr{C} = \emptyset$ then the algorithm terminates.
   Otherwise go back to Step 2

2D examples of the above algorithm are shown in Fig. 3.12. In Fig. 3.12, $\mathscr{P}_{\text{obj}}$ are depicted as white points and other input points are shown in black points. In the first state, (a), no object point is covered (marked as *uncovered* in Step 1). As a spherical support is generated in Step 2, some points are covered with the generated support and remarked as *covered*. In Fig. 3.12, a new support is drawn with a blue circle, while supports already existing are drawn in black. To avoid generating too many spherical supports, in the next iteration, new center is selected among uncovered-marked points, see (b) and (c). Finally all object points are covered with supports and a spherical cover is generated, (d).

Step 2 where some points are remarked as *covered*, is an expansion to 3D space of the 2D method proposed in [WK04]. In both methods, the object points to be remarked as *covered* are those inside the convex hull determined by the object points in the newly generated spherical support. See 2D examples in Fig. 3.12, such convex hulls are shown

in light blue and points which should be remarked are in blue. Note that points that are vertices of the convex hull are not considered as covered with the support. In the next iteration cycle, the center of a new spherical support will be selected among uncovered points. Using this method, generated spherical cover has no gap in most cases.

The number of generated spherical supports is much lower than the number of object points.

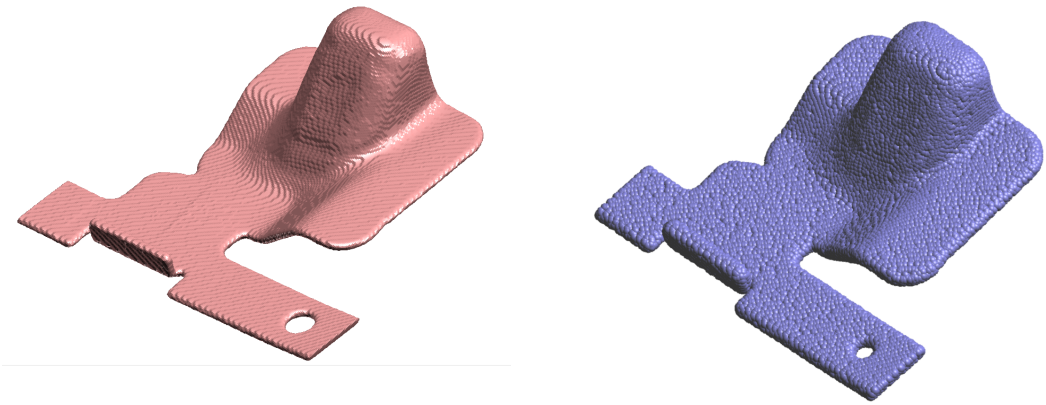


Fig. 3.11. Spherical cover for skeletal structure extraction.
Left: isosurface extracted from a CT-scanned data.
Right: spherical cover for the same object as that of the isosurface in the left image.

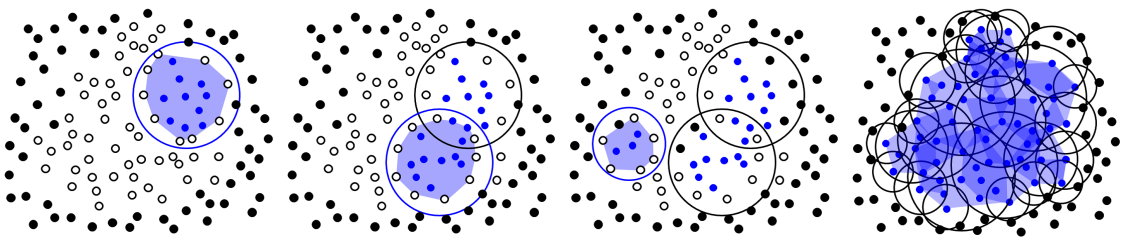


Fig. 3.12. Covering of object points (white points).
(a) A support is generated. Blue colored points mean covered points.
(b) and (c) Another support is generated.
(d) Final state. All object points are covered.

3.5 generation of Tetrahedral Mesh

After generating a spherical cover, we can generate a tetrahedral mesh as in Fig. 3.13 (as a triangular mesh in 2D). This tetrahedrization is realized with connecting the centers of spherical supports. The connectivity of tetrahedral mesh is determined using the adjacency of a spherical supports.

Tetrahedral mesh which our method adopts is a subset of weighted Delaunay tetrahedrization following the definition introduced by Edelsbrunner [Ede93]. Weighted Delaunay triangulation of $\mathscr{P}$ is defined by the dual of the weighted Voronoi diagram for $\mathscr{P}$ with a weighted distance rather than the Euclidean distance. Here is the weighted distance between a point $\boldsymbol{p}$ and a site $\boldsymbol{s}$ assigned a weight w_s

$$d_s(\boldsymbol{p}) = d(\boldsymbol{p}, \boldsymbol{s})^2 - w_s^2. \tag{3.13}$$

In our method, the centers of spherical supports work as sites and the radii of supports work as weights.

The connectivity of mesh can be determined on the basis of the adjacency of spherical supports. In the calculation of connectivity, the dimension of sphere is raised by a factor of one. That is, the adjacency of 3D/2D spheres is considered with 4D/3D spheres that have the same radii and centers as the original 3D/2D spheres. The triangulation obtained with this one-dimensional increase method is a subsets of weighted Delaunay triangulation. An example for a case changing from 2D to 3D is illustrated in Fig. 3.14. The left image shows an original 2D supports. Blue points mean the object points, and these are completely covered with supports. The right image shows the result of the one-dimensional increasing. 3D spheres and triangular mesh obtained with weighted Delaunay triangulation. Readers may notice that the tetrahedral mesh may not completely cover the object points near the boundary of the object. Our target structure exists near the center of object and tetrahedral mesh is used just for estimating the structure, and thus this uncovered region causes no problem.

Now let us consider a change from a 3D sphere to 4D as an example. To obtain the adjacency, checking the collision of boundaries of spherical supports is enough. This is the reason that the term sphere (the boundary of spherical support) is used in this section. The original 3D sphere centered at $(c_x, c_y, c_z) \in \mathbb{R}^3$ with radius r is described as $(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = r^2$. This sphere will become a 4D sphere with equation $(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 + (u - 0)^2 = r^2$. Note that the center of 4D sphere is the same point as the center of the original 3D sphere on the x, y, z space.

As mentioned above, the adjacency of 4D spheres determines the connectivity of tetrahe-

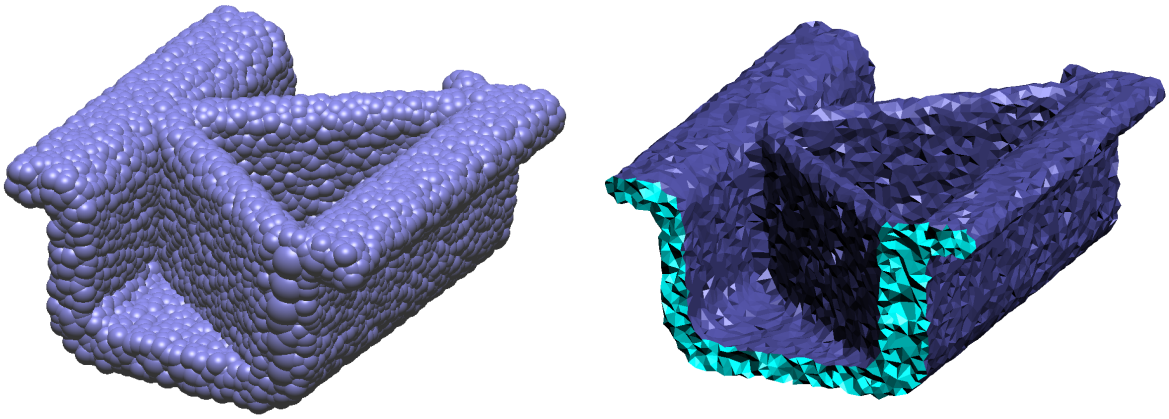


Fig. 3.13. Tetrahedrization using a spherical cover.
Left: a part of a spherical cover. Right: tetrahedral mesh generated from the spherical cover on the left.

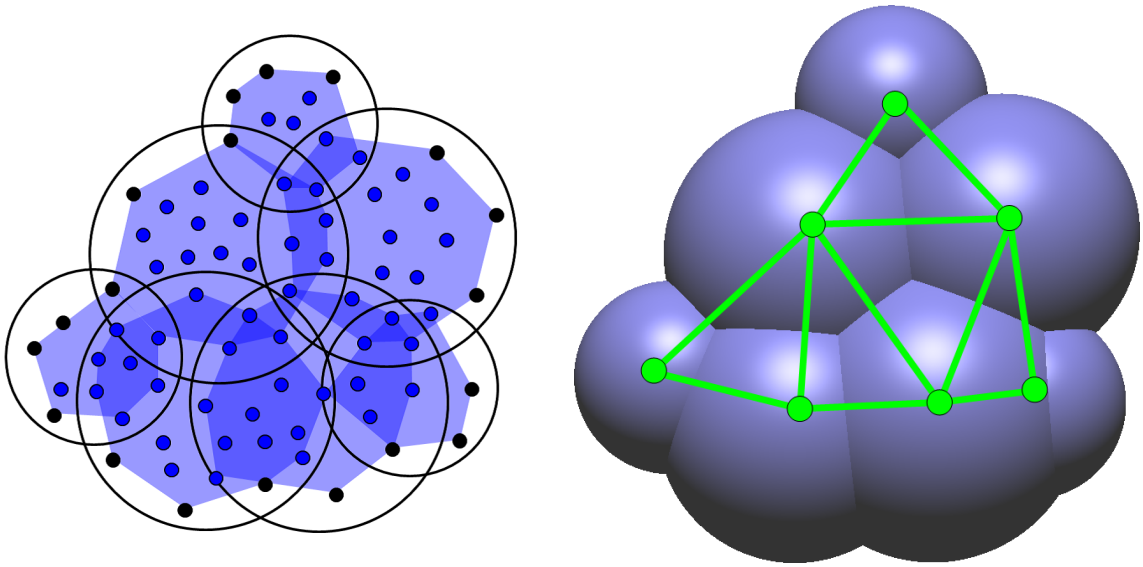


Fig. 3.14. Raising dimension of spherical supports from 2D to 3D.
Left: spherical supports for 2D points.
Right: 3D spheres generated with raising the dimension. Green lines shows the connectivity of triangular mesh obtained from these 3D spheres.

dral mesh. First, we select four 4D spheres with intersections. The number of intersection points are at most two points. Then if an intersection point is not included in other 4D spheres, the centers of these four 4D spheres are connected and a tetrahedron is created. As the centers of 4D spheres have a value of zero in the fourth coordinates, the generated tetrahedron exists in the 3D space. Examples of this triangulation in 2D are illustrated in Fig. 3.15. The triangulation process in 2D is basically same as 3D version. The only difference is that intersection points are considered for a set of three 3D spheres. The left images in Fig. 3.15 show original 2D spheres and the right images show 3D spheres. Yellow green points means the centers of spheres, and yellow points are the intersecting

points. Gray lines in the right images mean the x, y plane. The top column is a case that a triangle is generated while the bottom is not generated. In the latter case, the intersecting points are included by another sphere (depicted as a red circle).

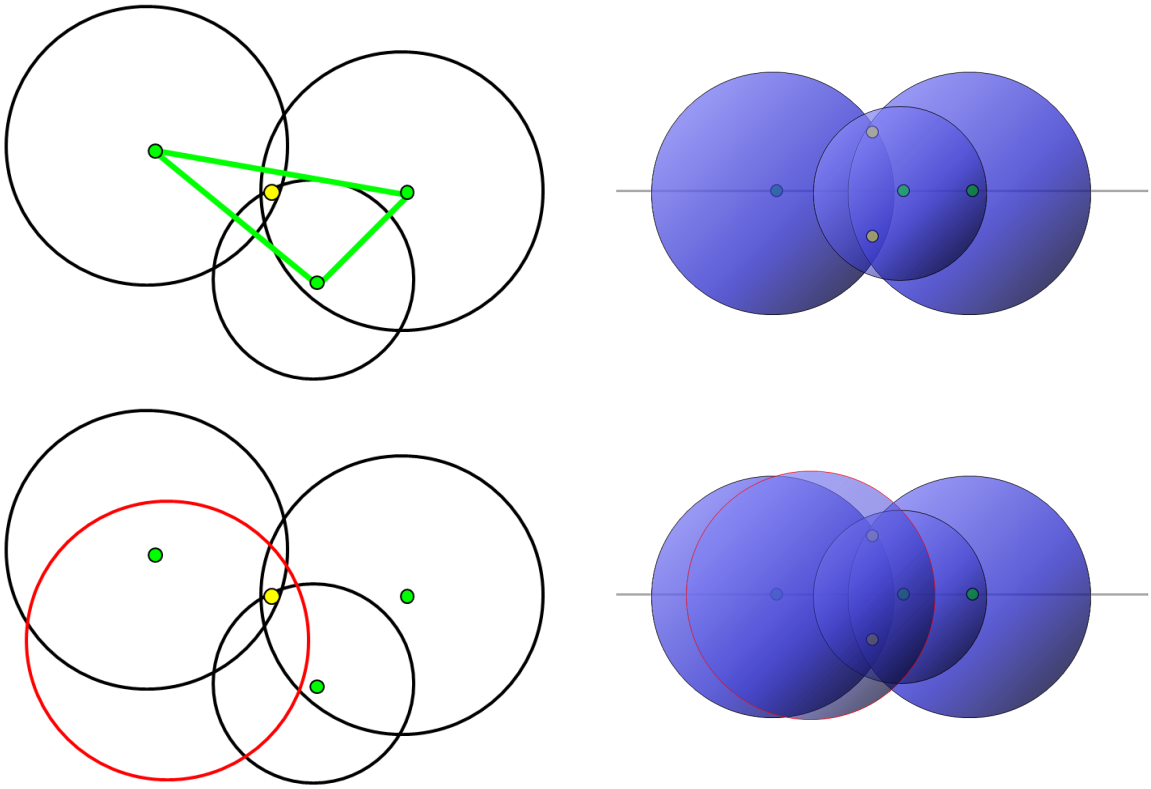


Fig. 3.15. Triangulation for 2D points.
Left: original 2D spheres. Right: 3D spheres (lateral view).
Top: case that a triangle is generated. Bottom: case that a triangle is not generated.

3.5.1 Avoiding Degeneracy

Let us consider the case that the input sampling points come from volumetric data. Typically volumetric data offers uniformly sampled points. As a result, in most cases, triangulations have a number of degenerate areas. To avoid this degeneracy, we slightly perturb the centers only in the intersection testing. Each center is moved $0.01 \times r_i$ in random direction. Note that the perturbed coordinates are used only for calculation of the connection. That it, the coordinates of vertices of obtained mesh are completely the same as ones before perturbation. This perturbation has no mathematical guarantee, but offers good results in this thesis.

3.5.2 Quality Improvement

The image on the left of Fig. 3.16 shows a 2D triangulation obtained applying the above strategy. In this triangulation, some skinny triangles (with a low aspect ratio) appear near the boundaries of the object. This is because some small spheres are absorbed by larger spheres and the centers of small spheres cannot appear in triangulation as vertices.

In terms of aspect ratio, the mesh quality can be improved through smoothing the size changes of triangles. Multiplying all radii of supports by $\alpha < 1$ and using these smaller radii facilitate the avoidance of intersections of small spheres being included in other spheres. This radius shrinking makes the number of vertices and tetrahedra of tetrahedral mesh increase. As a result, small tetrahedra appear near boundary and values of aspect ratio of tetrahedra increase. Thus the quality of tetrahedral mesh is improved. Good results are obtained using $\alpha = 0.75$ in all examples in this thesis. See the right-hand part of Fig. 3.16. Many small triangles appear near the boundary replacing the skinny ones, and thus the ratio of triangles with better values of aspect ratio increases.

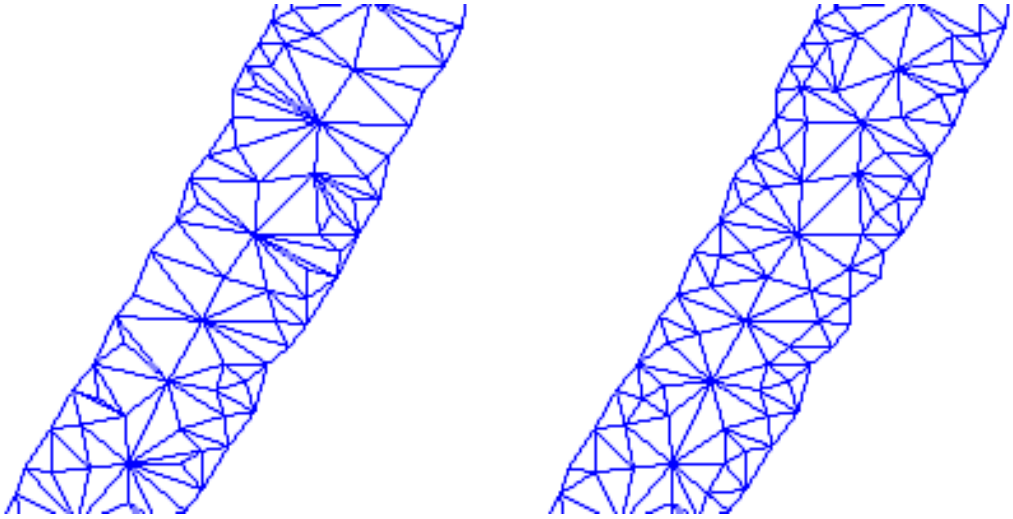


Fig. 3.16. Mesh quality improvement.
Left: triangulation without scaling of spherical supports.
Right: triangulation after scaling.

Fig. 3.13 is a result of this tetrahedrization for a part of a spherical cover of an industrial object. The number of all supports in the whole spherical cover is 44,095. The tetrahedral mesh corresponding to the whole spherical cover has 44,084 vertices and 226,474 tetrahedra.

3.6 Summary

In this chapter, the definition of spherical support and spherical cover are introduced and their data structure are described. They are followed by generation methods for sampling points from surfaces and for volumetric data. With the help of weighted Delaunay tetrahedrization, spherical covers can determine tetrahedral meshes. The specific algorithm of this spatial division is also explained.

In the following of this thesis we study enhancements and applications of this spherical covering technique. In the next Chapter 4, the differential operators on the spherical cover will be derived. Some algorithms applied spherical cover are proposed in the other following chapters.

Chapter 4

Differential Operators on Sphere Covering

4.1 Introduction

Differential operators are fundamental mathematical tools and appear in many geometry processing applications involving smoothing and shape analysis using curvature. Some shape representations have differential operators which are specialized to their data structure.

Unfortunately, spherical covers have no proper differential operators. So in this chapter, we define new differential operators for a spherical cover using the set of supports as a grid-like data structure. We show only the 3D case in the following and discuss about the discretization of the differential operators in nD space in Section 4.5 with showing expected formulas.

4.2 Differential Operators

Before deriving new differential operators, definitions of the continuous differential operators are presented.

The **nabla** is a vector operator and also called **Del**. This operator is described in Cartesian coordinate system $\mathbb{R}^3$ as

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right). \tag{4.1}$$

The **gradient** of a scalar field f is a vector field represented as

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right). \tag{4.2}$$

The **divergence** of a vector field $\boldsymbol{v} = (v_x, v_y, v_z)$ is the dot product of ∇ with $\boldsymbol{v}$

$$\nabla \cdot \boldsymbol{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}. \tag{4.3}$$

The **Laplacian operator** is defined as

$$\triangle \equiv \nabla^2 = \nabla \cdot \nabla = \frac{\partial}{\partial x^2} + \frac{\partial}{\partial y^2} + \frac{\partial}{\partial z^2}. \tag{4.4}$$

There is also a vector function **curl** defined as $\nabla \times \boldsymbol{v}$, however this thesis does not handle this function.

4.3 Derivation

In most applications on a spherical cover, operations are performed only at the centers of spherical supports. So we define new discrete operators of their centers. To distinguish our discrete operators from traditional continuous ones, hereafter we denote new operators as Grad$(\cdot)$, Div$(\cdot)$ and Lap$(\cdot)$ corresponding to the traditional $\nabla(\cdot)$, $\nabla \cdot (\cdot)$ and $\triangle(\cdot)$, respectively.

First of all, we assume that our gradient field satisfies the following condition:

Assumption 4.3.1

Each spherical support s_i is assigned a constant vector $\boldsymbol{v}_i$.

Under the assumption 4.3.1, we construct a vector field in a PU manner. The vector at arbitrary point $\boldsymbol{x}$ is evaluated as follows:

$$\boldsymbol{v}(\boldsymbol{x}) = \frac{\sum_i w_i(\boldsymbol{x})\boldsymbol{v}_i}{\sum_i w_i(\boldsymbol{x})}. \tag{4.5}$$

Now we start operator derivation. First, the divergence operator Div$(\boldsymbol{v}_i)$ will be derived. We define Div$(\boldsymbol{v}_i)$ as the average of integration of the continuous divergence $\nabla \cdot \boldsymbol{v}(\boldsymbol{x})$ inside s_i;

$$\frac{4\pi r_i^3}{3}\mathrm{Div}\,(\boldsymbol{v}_i) \equiv \int_{\boldsymbol{x}\in s_i} \nabla \cdot \boldsymbol{v}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{4.6}$$

Here we introduce the divergence theorem for each support.

Divergence theorem

Let V be a region with boundary S. Then the volume integral of the divergence of the vector field $\boldsymbol{v}$ over V is equal to the surface integral of $\boldsymbol{v}$ over S;

$$\int_{\boldsymbol{x}\in V} (\nabla \cdot \boldsymbol{v})\mathrm{d}V = \int_{\boldsymbol{x}\in S} \boldsymbol{v} \cdot \boldsymbol{n}\mathrm{d}S. \tag{4.7}$$

Following the divergence theorem, the right hand side of equation (4.6) can be represented as a surface integral over the boundary ∂s_i of s_i:

$$\frac{4\pi r_i^3}{3}\mathrm{Div}\,(\boldsymbol{v}_i) = \int_{\boldsymbol{x}\in \partial s_i} \boldsymbol{v}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \tag{4.8}$$

where $\boldsymbol{n}(\boldsymbol{x})$ is the unit normal at point $\boldsymbol{x}$ on ∂s_i.

We discretize this surface integral with surface integrals inside neighbors $\{s_j\}$ of s_i. Let $\delta_{i,j}$ be the part of ∂s_i inside a neighbor s_j. See Fig. 4.1, an illustration of 2D spherical supports. In Fig. 4.1, $\delta_{i,j}$ is indicated by the green dotted line.

For simplicity, we introduce the following assumption.

Assumption 4.3.2

$\delta_{i,j}$ is covered only by s_i and s_j.

Under Assumption 4.3.2, $\boldsymbol{v}(\boldsymbol{x})$ in $\delta_{i,j}$ is $\boldsymbol{v}_j$. Now, the approximation of the left hand side of equation (4.8) is approximated by a sum of surface integrals over $\delta_{i,j}$:

$$\frac{4\pi r_i^3}{3}\mathrm{Div}(\boldsymbol{v}_i) \approx \frac{4\pi r_i^2}{\sum_j \int_{\boldsymbol{x}\in\delta_{i,j}} d\boldsymbol{x}} \sum_j \int_{\boldsymbol{x}\in\delta_{i,j}} \boldsymbol{v}_j \cdot \boldsymbol{n}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{4.9}$$

When neighbors of s_i overlap each other, the sum of the areas of $\{\delta_{i,j}\}$ is greater than the true value $4\pi r_i^2$ of the area of ∂s_i. In the approximation (4.9), denominator $\sum_j \int_{\boldsymbol{x}\in\delta_{i,j}} \mathrm{d}\boldsymbol{x}$ on the right hand side works as a normalizer.

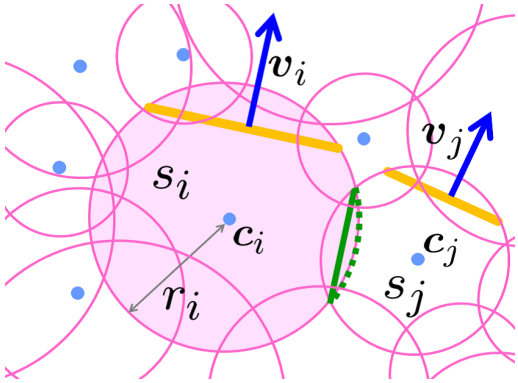


Fig. 4.1. Spherical supports in 2D. The orange lines are isolines of the local approximations, and the blue arrows are their normals. $\delta_{i,j}$ is indicated by the green dotted line, and the intersection disk with area $D_{i,j}$ is indicated by the solid green line.

Discretizing the integrals of (4.9) provides a new divergence operator at $\boldsymbol{c}_i$. Following is the way to discretization. Let us start with the right hand side of (4.9). Calculation is performed on spherical-coordinate system

$$\begin{cases} x = R\sin\theta\cos\varphi \\ y = R\sin\theta\sin\varphi \\ z = R\cos\theta \end{cases} \tag{4.10}$$

where $\theta \in [-\frac{1}{2}\pi, \frac{1}{2}\pi]$ and $\varphi \in [0, 2\pi]$, see 4.2. The origin of this coordinate system is the center $\boldsymbol{c}_i$. The center $\boldsymbol{c}_j$ is on the z-axis, that is, the centers $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ are on the $x-y$

plane. The Jacobian matrix of a function $f(\boldsymbol{x})$ is described as

$$
\begin{aligned}
Jf(\boldsymbol{x}) &= \begin{pmatrix} \frac{\partial x}{\partial R} & \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial \varphi} \\ \frac{\partial y}{\partial R} & \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial \varphi} \\ \frac{\partial z}{\partial R} & \frac{\partial z}{\partial \theta} & \frac{\partial z}{\partial \varphi} \end{pmatrix} \\
&= \begin{pmatrix} \sin\theta\cos\varphi & R\cos\theta\cos\varphi & -R\sin\theta\sin\varphi \\ \sin\theta\sin\varphi & R\cos\theta\sin\varphi & R\sin\theta\cos\varphi \\ \cos\theta & -R\sin\theta & 0 \end{pmatrix}.
\end{aligned} \tag{4.11}
$$

The Jacobian determinant J is

$$
\begin{aligned}
J &= \det\left(Jf(\boldsymbol{x})\right) \\
&= R^2 \sin\theta.
\end{aligned} \tag{4.12}
$$

The normal vector $\boldsymbol{n}(\boldsymbol{x})$ at a point $\boldsymbol{x}$ on boundary ∂s_i is represented as

$$
\boldsymbol{n}(\boldsymbol{x}) = \begin{pmatrix} \sin\theta\cos\varphi \\ \sin\theta\sin\varphi \\ \cos\theta \end{pmatrix}. \tag{4.13}
$$

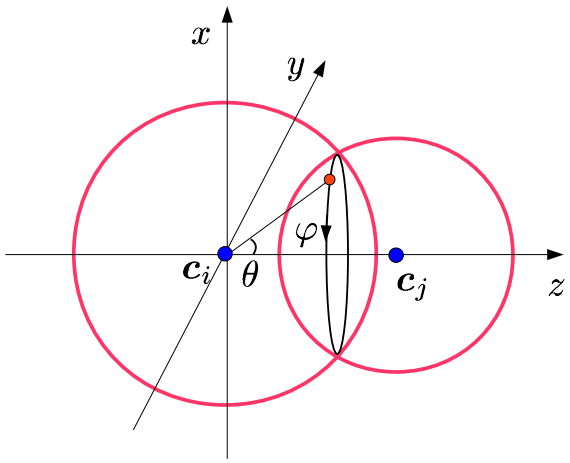


Fig. 4.2. Two spherical supports in spherical-coordinate system.

Let the gradient vector $\boldsymbol{v}_j$ be

$$
\boldsymbol{v}_j = \begin{pmatrix} v_{j_x} \\ v_{j_y} \\ v_{j_z} \end{pmatrix}. \tag{4.14}
$$

The surface integral $\int_{\boldsymbol{x}\in\delta_{i,j}} \boldsymbol{v}_j \cdot \boldsymbol{n}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ in approximation (4.9) is calculated as follows:

$$
\begin{aligned}
\int_{\boldsymbol{x}\in\delta_{i,j}} \boldsymbol{v}_j \cdot \boldsymbol{n}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} &= \int_0^{2\pi}\int_0^{\theta_j} \boldsymbol{v}_j \cdot \boldsymbol{n}(\boldsymbol{x}) R^2 \sin\theta d\theta d\varphi \\
&= \int_0^{2\pi}\int_0^{\theta_j} \left(v_{j_x}\sin\theta\cos\varphi + v_{j_y}\sin\theta\sin\varphi + v_{j_z}\cos\theta\right) R^2\sin\theta d\theta d\varphi \\
&= \pi\left(R\sin\theta_j\right)^2 v_{j_z}.
\end{aligned} \tag{4.15}
$$

$\pi\left(R\sin\theta_j\right)^2$ is equal to the area, described as $D_{i,j}$, of the intersection disk of supports s_i and s_j. This suggest that the integral of continuous divergence over $\delta_{i,j}$ is equal to the integral of inner product of the assigned vector $\boldsymbol{v}$ and the normal over corresponding intersection disk. Remaining v_{j_z} is caused by the symmetric property of $\boldsymbol{n}(\boldsymbol{x})$ about φ in the integral range. In addition, from the symmetric property of $\boldsymbol{n}(\boldsymbol{x})$ in the integral range,

$$v_{j_z} = \boldsymbol{v}_j \cdot \frac{\boldsymbol{c}_j - \boldsymbol{c}_i}{\|\boldsymbol{c}_j - \boldsymbol{c}_i\|} \tag{4.16}$$

holds. Therefore, the right hand side of (4.9) can be discretized as follows:

$$\frac{4\pi r_i^2}{\sum_j \int_{\boldsymbol{x}\in\delta_{i,j}} d\boldsymbol{x}} \sum\nolimits_j \int_{\boldsymbol{x}\in\delta_{i,j}} \boldsymbol{v}_j \cdot \boldsymbol{n}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \frac{4\pi r_i^2 \sum_j D_{i,j}}{\sum_j \int_{\boldsymbol{x}\in\delta_{i,j}} d\boldsymbol{x}} \boldsymbol{v}_j \cdot \frac{\boldsymbol{c}_j - \boldsymbol{c}_i}{\|\boldsymbol{c}_j - \boldsymbol{c}_i\|}. \tag{4.17}$$

Following the above calculations, a discrete divergence operator is obtained:

$$\mathrm{Div}(\boldsymbol{v}_i) = \frac{3}{r_i S_i} \sum\nolimits_j D_{i,j}\ \boldsymbol{v}_j \cdot \frac{\boldsymbol{c}_j - \boldsymbol{c}_i}{\|\boldsymbol{c}_j - \boldsymbol{c}_i\|}, \tag{4.18}$$

$$S_i = \sum\nolimits_j \int_{\boldsymbol{x}\in\delta_{i,j}} d\boldsymbol{x}. \tag{4.19}$$

For simpler notation, introducing

$$\phi_{i,j} = \frac{D_{i,j}}{\|\boldsymbol{c}_j - \boldsymbol{c}_i\|}, \tag{4.20}$$

equation (4.18) can be rewritten as

$$\mathrm{Div}(\boldsymbol{v}_i) = \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j} \boldsymbol{v}_j \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i). \tag{4.21}$$

Now a discrete divergence operator has been defined.

Next, we define the discrete gradient operator of a vector field with the first-order Taylor expansion at $\boldsymbol{c}_j$:

$$\boldsymbol{v}_i \approx \boldsymbol{v}_j + \mathrm{Grad}(\boldsymbol{v}_j) \cdot (\boldsymbol{c}_i - \boldsymbol{c}_j). \tag{4.22}$$

Now we have the divergence and gradient operators, so the discrete Laplacian operator will be derived using these operators. It can be derived through

$$\mathrm{Lap}(\boldsymbol{v}_i) = \mathrm{Div}(\mathrm{Grad}(\boldsymbol{v}_i)) \tag{4.23}$$

in a way similar to the continuous form

$$\triangle(\boldsymbol{v}_i) = \nabla \cdot \nabla(\boldsymbol{v}_i). \tag{4.24}$$

With replacing $\boldsymbol{v}$ with $\mathrm{Grad}(\boldsymbol{v})$ in the definition of discrete divergence operator (4.18), the right hand side of equation 4.23 is obtained;

$$\mathrm{Div}(\mathrm{Grad}(\boldsymbol{v}_i)) = \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j} \mathrm{Grad}(\boldsymbol{v}_j) \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i). \tag{4.25}$$

Using (4.22), the inner product in (4.25) can be replaced by the difference of vector $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$:

$$\mathrm{Div}(\mathrm{Grad}(\boldsymbol{v}_i)) = \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j}(\boldsymbol{v}_j - \boldsymbol{v}_i). \tag{4.26}$$

Finally the discrete Laplacian operator is defined:

$$\mathrm{Lap}(\boldsymbol{v}_i) \equiv \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j}(\boldsymbol{v}_j - \boldsymbol{v}_i). \tag{4.27}$$

The above operator is the Laplacian of a vector field. In the same manner, the Laplacian on a scalar field with a linear approximation function $f(\boldsymbol{c}_i)$ can also be derived through

$$\mathrm{Lap}(f(\boldsymbol{c}_i)) = \mathrm{Div}(\mathrm{Grad}(f(\boldsymbol{c}_i))). \tag{4.28}$$

Note that $f(\boldsymbol{x})$ must be a linear function so that $\mathrm{Grad}(f(\boldsymbol{x}))$ becomes a constant vector, because our operator $\mathrm{Div}(\cdot)$ is only for a constant vector. The Laplacian of $f(\boldsymbol{c}_i)$ is

$$\mathrm{Lap}(f(\boldsymbol{c}_i)) \equiv \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j}(f(\boldsymbol{c}_j) - f(\boldsymbol{c}_i)). \tag{4.29}$$

If $f(\boldsymbol{x})$ is offered as weighted average of linear local approximations, $f(\boldsymbol{x})$ has an $O(r^2)$ approximation order (r is the support size of a local approximation) to the original surface.

4.4 Implemental Calculation

In this section, arithmetic details of discrete differential operators are shown.

Again consider on the spherical-coordinate system. We rewrite approximation (4.9) with terms of spherical-coordinate system. See Fig. 4.3, a support s_i and its neighbors. First, the surface integration over $\delta_{i,j}$ is calculate as

$$\begin{aligned} \int_{\boldsymbol{x}\in\delta_{i,j}} d\boldsymbol{r} &= \int_0^{2\pi} \int_0^{\theta_j} r_i^2 \sin\theta \, d\theta \, d\phi \\ &= 2\pi r_i^2 \left(1 - \cos\theta_j\right). \end{aligned} \tag{4.30}$$

Following this result and the fact that the surface integration of continuous divergence of a vector over $\delta_{i,j}$ is equal to the integral of the inner product of the vector and the normal over an intersection disk of s_i and s_j, the approximation (4.9) can be calculated as follows:

$$\begin{aligned} \frac{4\pi r_i^3}{3} \nabla \cdot \boldsymbol{v}(\boldsymbol{x}) &\approx \frac{4\pi r_i^2}{\sum_j \int_{\boldsymbol{x}\in\delta_{i,j}} d\boldsymbol{x}} \sum\nolimits_j \int_{\boldsymbol{x}\in\delta_{i,j}} \boldsymbol{v}_j \cdot \boldsymbol{n}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \\ &= \frac{4\pi r_i^2 \sum_j \pi \left(r_i \sin\theta_j\right)^2 v_{j_z}}{\sum_j 2\pi r_i^2 \left(1 - \cos\theta_j\right)} \\ &= \frac{2\pi \sum_j \left(r_i \sin\theta_j\right)^2 v_{j_z}}{\sum_j \left(1 - \cos\theta_j\right)}. \end{aligned} \tag{4.31}$$

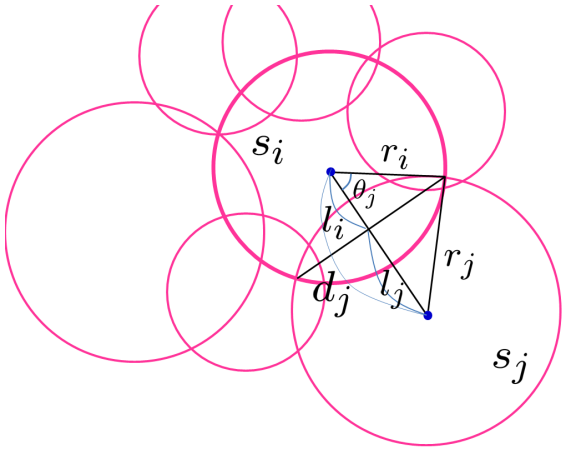


Fig. 4.3. Support s_i and its neighbors.

Expression of discrete divergence operator in spherical-coordinate system is led by the above calculation to be

$$\mathrm{Div}(\boldsymbol{v}_i) \approx \frac{3\sum_j (r_i \sin\theta_j)^2 \, v_{j_z}}{2r_i^3 \sum_j (1-\cos\theta_j)}. \tag{4.32}$$

The values of $r_i \sin\theta_j$ and $r_j \cos\theta_j$ can be obtained through elementary geometrical calculation. Let us consider a case that two supports s_i and s_j with radii r_i and r_j intersect each other. The distance between the centers are d_j. It is decomposed to l_i and l_j at the crossing point with the intersection disk, see Fig. 4.3. In this case, it is observed that $r_i \sin\theta_j$ and $\cos\theta_j$ become

$$r_i \sin\theta_j = \sqrt{r_i^2 - l_i^2}, \tag{4.33}$$

$$\cos\theta_j = \frac{l_i}{r_i}. \tag{4.34}$$

In addition, the following two equations holds:

$$\begin{cases} r_i^2 - l_i^2 = r_j^2 - l_j^2 \\ d_j = l_i + l_j \end{cases}. \tag{4.35}$$

These relations suggests that the value of l_i is obtained as

$$l_i = \frac{d_j^2 + r_i^2 - r_j^2}{2d_j}. \tag{4.36}$$

Consequently, the spherical-coordinate expression of $\mathrm{Div}(\boldsymbol{v}_i)$ becomes as follows:

$$\mathrm{Div}(\boldsymbol{v}_i) = \frac{3}{2r_i^2} \frac{\sum_j \left(r_i^2 - \left(\frac{d_j^2 + r_i^2 - r_j^2}{2d_j} \right)^2 \right) v_{j_z}}{\sum_j \left(r_i - \frac{d_j^2 + r_i^2 - r_j^2}{2d_j} \right)}. \tag{4.37}$$

4.5 Discussion

The new discrete differential operators derived in this chapter are operators only for a constant vector. Moreover, the new discrete Laplacian operator for a function assumes its argument is a linear function. These assumptions are very strong simplifications, so more verification may be needed.

In geometry processing area, many applications use linear approximations. Hence assuming the local approximation functions are linear is reasonable.

Nevertheless, changing the primitive functions to be quadratic and cubic functions can extend their application areas. This change can be easily achieved with straightforward extension of calculation, but the calculation is quite complicated.

Another problem is the asymmetry in the operators of two intersecting spheres. This problems can be solved using a unique area partitioning such as the Laguere Voronoi diagram. Specifying the area partition with such a method is our future work.

We defined the discrete gradient, divergence, and Laplacian operators of a vector field, but leave the curl. Derivation of the curl of a vector field is a worthful future work to be performed.

Another issue is the derivation of the discrete differential operators in nD. With the formulas of the discrete operators in 3D, the nD-discretized operators are expected to be

$$\boldsymbol{v}_i \approx \boldsymbol{v}_j + \mathrm{Grad}(\boldsymbol{v}_j) \cdot (\boldsymbol{c}_i - \boldsymbol{c}_j) \tag{4.38}$$

$$\mathrm{Div}(\boldsymbol{v}_i) \equiv \frac{n}{r_i S_i} \sum\nolimits_j \phi_{i,j} \boldsymbol{v}_j \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i) \tag{4.39}$$

$$\mathrm{Lap}(\boldsymbol{v}_i) \equiv \frac{n}{r_i S_i} \sum\nolimits_j \phi_{i,j} (\boldsymbol{v}_j - \boldsymbol{v}_i) \tag{4.40}$$

$$\mathrm{Lap}(f(\boldsymbol{c}_i)) \equiv \frac{n}{r_i S_i} \sum\nolimits_j \phi_{i,j} (f(\boldsymbol{c}_j) - f(\boldsymbol{c}_i)) . \tag{4.41}$$

Proving higher dimensional case ($n \geq 3$) is not obvious while the 2D case can be proved straightforward changing the derivation of the 3D case.

The set of the proposed discrete differential operators can be applied to many applications such as smoothing vectors and volume processing. In the latter part of this thesis, we show two instances of applications of the discrete differential operators.

4.6 Summary

In this chapter, we derived new differential operators for a spherically covered domain under two assumptions:

- Each spherical support s_i is assigned a constant vector $\boldsymbol{v}_i$,
- the region $\delta_{i,j}$ of boundary of a spherical support s_i inside another support s_j is covered only by s_i and s_j.

Our differential operators are gradient, divergence, and Laplacian operators on a constant vector, and Laplacian operator on a scalar. These operators are defined at centers of spherical supports.

Chapter 5

Surface Reconstruction

5.1 Introduction

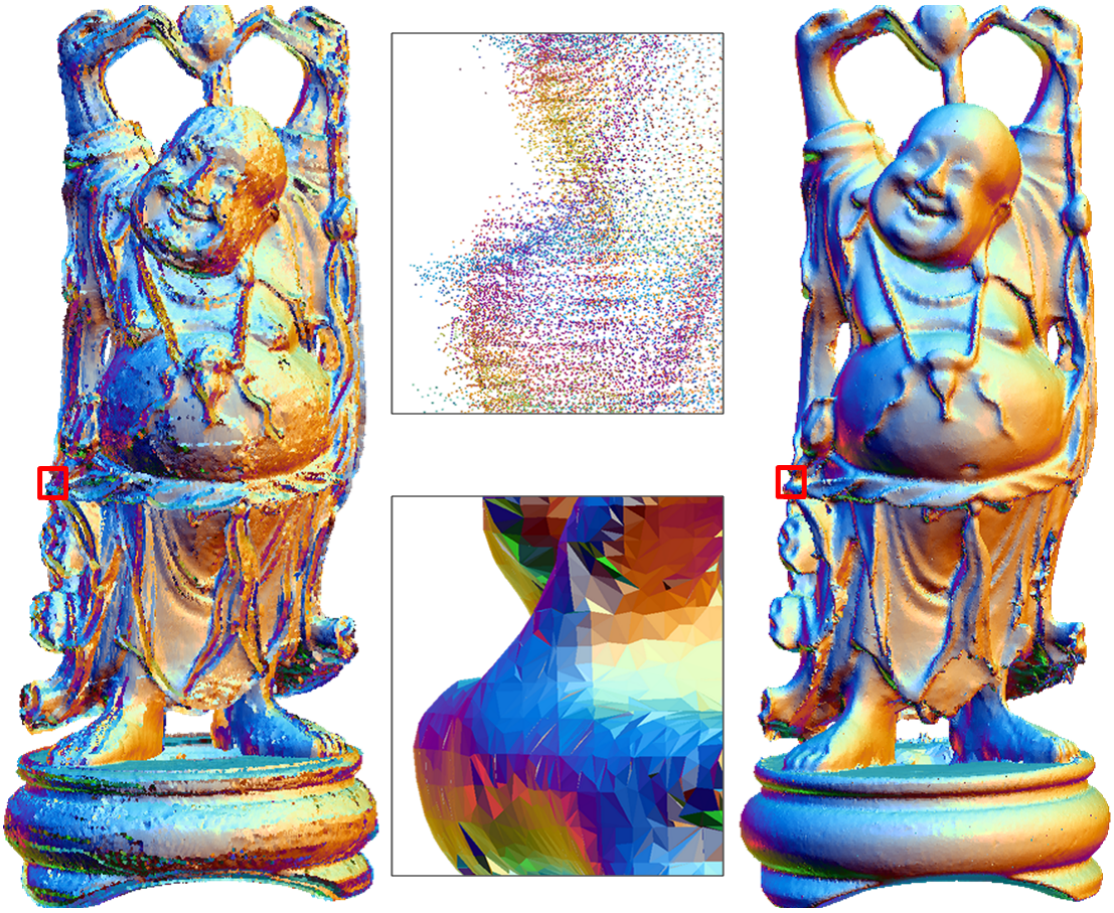


Fig. 5.1. Examples of sampling points and a surface mesh.
Left: a set of sampling points obtained using a laser scanner.
Middle: close-ups of boxed parts of the sampling points and the surface mesh.
Right: reconstructed surface mesh.

Surface reconstruction is a process generating a surface from sampling points. Sometimes, if necessary, generated surfaces are approximated by surface meshes. This subject

is a very important in many industrial areas where shapes of real objects are handled, including mechanical engineering as CAD, CAE and computer graphics. Surface reconstruction is an essentially hard issue in the point that original shape cannot be completely known from sampling points. Moreover, as mentioned in section 2.1.1, some kinds of problems such as noise, outliers, and lack of sampling make surface reconstruction difficult. In order to overcome these difficulties, many algorithms are proposed. Some methods related in this work are introduced in section 2.2.

In this section, a surface reconstruction algorithms is proposed on the basis of partition of unity approach. It is robust for low-quality data because of using Graph-cut and diffusion technique.

5.1.1 Problem Setting

The goal of surface reconstruction is generating a surface representing an object from sampling points from the surface of the object. Generated surface can be convert to a 2-manifold triangular mesh which may have boundaries. One example of reconstruction from scanned points is shown in Fig. 5.1. Sampling points can be obtained by use of scanning devices as denoted in section 2.1.1.

In this work, sampling points are assumed to have oriented normals. If normals are not equipped, we can obtain those with a simple method, for instance, evaluation based on the principal component analysis as introduced in [HDD+92].

5.1.2 Our Approach

We are led by the discussions in section 2.2.2 to propose a novel surface reconstruction algorithm that is essentially based on a local implicit method. Local approach has advantage to represent details. But the existing local algorithms are fitting-only approaches, so cannot handle low quality data well. Therefore generating a smooth surface only with a local method was a very difficult problem.

To achieve a smooth and precise representation, we combine a local method and a global method, and then smooth the local approximations using a diffusion technique. We adopt a PU implicit as a local implicit method, Graph-cut as a global, and Laplacian smoothing as a smoothing method. Fig. 5.2 shows the overview of our algorithm. First, the initial spherical cover is generated with PU approach. Then Graph-cut is performed. The purpose of the second step is removing the affects of outliers. The third step is smoothing of local approximation functions. The affects of noise are removed in this step. Finally, reconstructed surface is converted to a surface mesh.

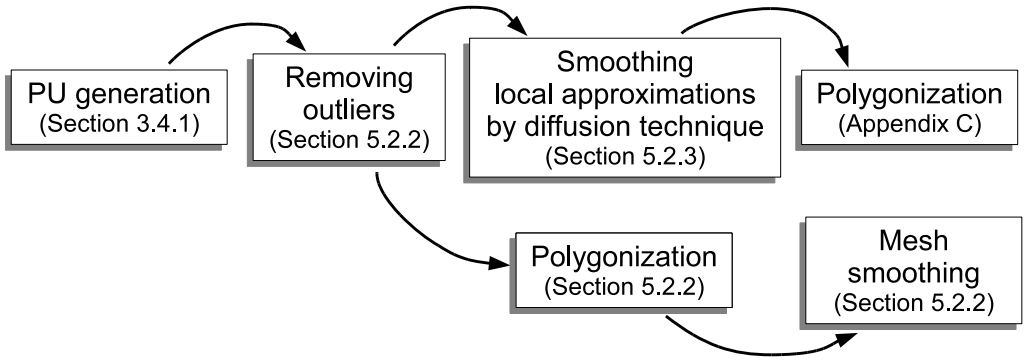


Fig. 5.2. Overview of surface reconstruction algorithm.

Outlier Robustness. PU is a local approximation technique, meaning that obtained surfaces have high accuracy but are sensitive to noise. Graph-cut, on the other hand, is a global algorithm that is robust to noise but produces low-accuracy results because it is a discrete binary operation. With combining these two approaches, we can find spherical supports which are affected by outliers and lack of samplings. Fig. 5.3 shows the effect of this combination. While the result obtained using only PU has many undesired surfaces, result provided by proposed combination method has no such surfaces.

Fig. 5.3. Surface reconstruction for sampling points with many outliers.
Left: sampling points including many outliers.
Middle: result without Graph-cut.
Right: result with Graph-cut.

Noise Robustness. For the purpose of improving noise-robustness, our aim is to smooth the local approximation functions using the differential operators derived in Chapter 4.

This smoothing inherits the benefits of local implicit methods and smoothing.

We perform the Laplacian smoothing for a vector field defined by the gradient of the approximation function. Then the approximation functions are updated to reflect the smoothing of the gradient field. This smoothing process achieves a strong noise-robustness, see Fig. 5.4. The bumps in the surface obtained from the initial PU do not appear in the surface generated using our smoothing method.

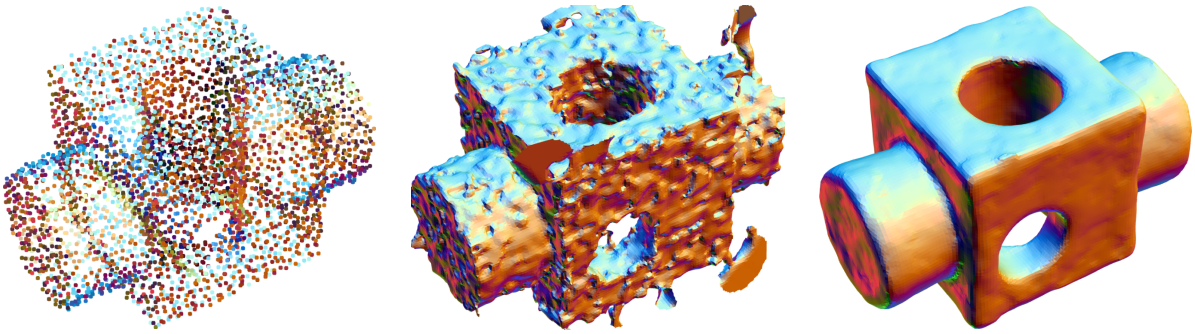


Fig. 5.4. Surface reconstruction for sampling points with noise.
Left: sampling points including noise.
Middle: result without smoothing of local approximation functions.
Right: results with smoothing.

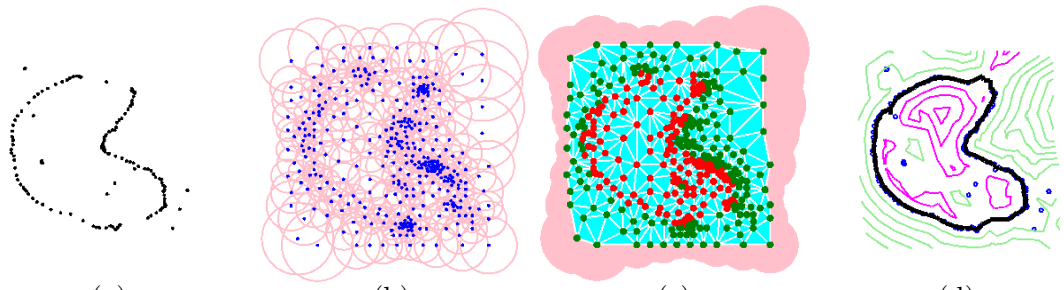
5.2 Algorithm

5.2.1 Overview

Here we overview the whole algorithm with 2D-version examples in Fig. 5.5. As input, we take sampling points equipped with oriented normals. We make no assumption as to the quality of sampling points; input possibly includes noise, outliers and areas with a lack of sampling as Fig. 5.3 and 5.4. The alphabets in the last of sentences correspond to the images in Fig. 5.5.

Algorithm: surface reconstruction

1. Generate a spherical cover for the bonding box through Partition of Unity (b).
2. Generate a tetrahedral mesh. Classify the mesh vertices into inside and outside of the object by Graph-cut (c).
3. Smooth local approximation functions by diffusion approach.
4. Generate a surface mesh by polygonizing the zero-level set (d).



(a) (b) (c) (d)

Fig. 5.5. Processes of the proposed surface reconstruction algorithm in 2D.
(a) Sampling points.
(b) Generated spherical cover.
(c) The results of Graph-cut; the red points are judged as being inside the object, while the green ones are deemed to be outside. A part of the graph is depicted in white.
(d) Reconstructed surface (black line). The contours of $f(\boldsymbol{x})$ are drawn with pink and green lines according to the signs of $f(\boldsymbol{x})$. Blue points mean sampling points.

First, a spherical cover whose local approximations are linear polynomials is constructed for a bounding box of a point cloud. Then a PU implicit function whose zero-level set represents the shape of an object can be obtained. How to generate a spherical cover has already been explained in Section 3.4.1.

After generating a PU implicit function, Graph-cut is performed to separate the covered domain into inside and outside areas of the object. The aim of step is recognition of spherical supports which may be affected by outliers and lack of sampling. The details are described in Section 5.2.2.

Next, smoothing of local approximation functions is performed by diffusion on a spherical cover. In this step, the differential operators derived in Chapter 4 are used. This smoothing is explained in Section 5.2.3.

Finally, the isosurface with value zero is extracted and polygonized. The polygonization method adopted in this step is an existing and common one. See Appendix C for more details.

The integration of Graph-cut approach and the diffusion approach is explained in Section 5.2.4.

5.2.2 Graph-cut

Introduction

Since PU is based on a local fitting, it achieves high-accuracy representation of objects but very sensitive to low-quality data involving outliers and lack of sampling. Our aim is combining local approach and a global algorithm and develop a more stable al-

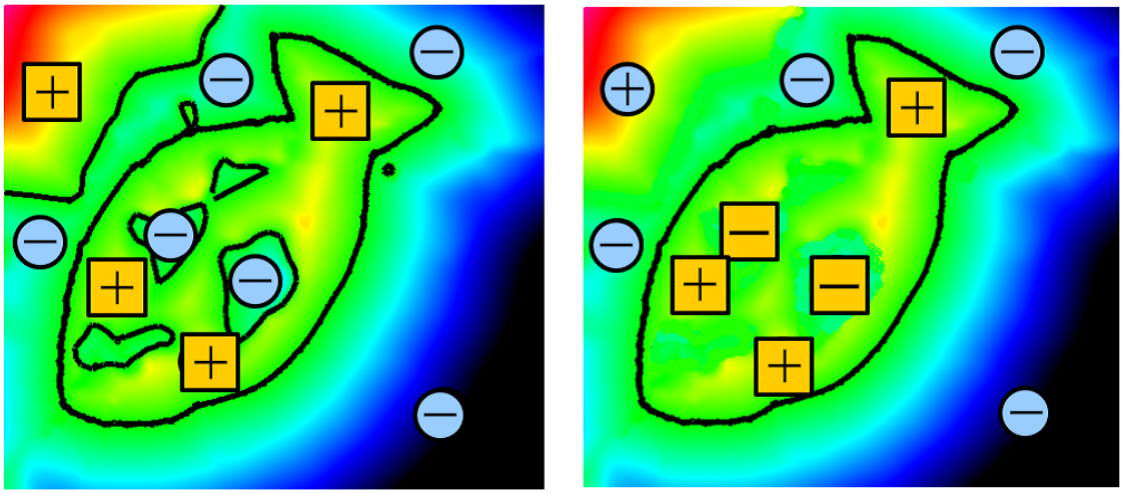


Fig. 5.6. Inside/outside classification.
Left: classification only through partition of unity.
Right: classification through combination of partition of unity and Graph-cut.

gorithm. Several global algorithms have been proposed such as Graph-cut [BK04] or FEM approach [SLS+07], among them, we adopt Graph-cut as a global method rather than FEM approach. The reason is, we have already a local approximation and just a global inside/outside classification is enough for outlier removal. In addition, Graph-cut is practically faster and more memory efficient than FEM approach. The effects of this combination for inside/outside classification are shown in Fig. 5.6, the comparison of results only with PU approach with results obtained using the combination of PU and Graph-cut. In both images, the values of approximation function $f(\boldsymbol{x})$ are indicated with colors red for high value and blue for low value. Reconstructed surfaces are shown with black lines. Points judged as inside of an object are indicated with yellow square points while points judged as outside are indicated with blue round points. The sign inside each point means the sign of $f(\boldsymbol{x})$ at the point. Result obtained only with partition of unity has extra surfaces because of the locality of the partition of unity. The globalness of Graph-cut helps to overcome this problem as observed in the right image of Fig. 5.6.

Another results for 3D sampling points are shown in Fig. 5.3. The input (the left image) is very noisy data. While the PU-only method fails to detect an appropriate surface affected by many outliers (the middle image), global-local combination achieves outlier-robust surface reconstruction (the right image).

Algorithm

The aim of this step is, performing Graph-cut to determine whether the support centers are inside or outside of the object. The detail of Graph-cut is described in Appendix A.

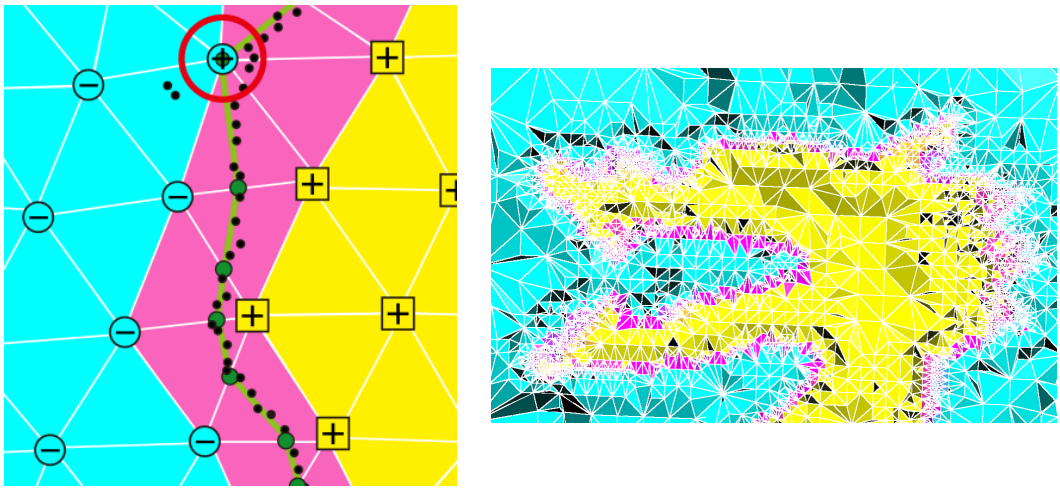


Fig. 5.7. Result of classification using Graph-cut.
Left: a 2D-version result. Surface is depicted with yellow green line.
Right: a cross section of a tetrahedral mesh for the head of the Stanford dragon.

The support centers work as vertices in our graph. Thus the support center classification is equal to the vertex classification in this step. Following the vertex classification, we can also classify the tetrahedra. Fig. 5.7 shows the classification of tetrahedra. The image on the left is a 2D-version classification. Black points are sampling points. Blue round points indicate outside vertices, while yellow squared ones are inside vertices. The signs inside points are signs of the approximation function $f(\boldsymbol{x})$ at the points. Yellow tetrahedra have four inside vertices, blue have four outside vertices, and pink have both types of vertices. The surface is considered to exist inside the pink tetrahedra. In the left image, the surface is depicted in a yellow green line. The red-circled point in the left image is regarded as existing outside following the classification through Graph-cut although the sign of $f(\boldsymbol{x})$ claims this point is inside. The image on the right shows a cross section of a resulting tetrahedral mesh of the Stanford dragon. The coloring rules follow the left image.

In the first of all, a graph on which Graph-cut is performed should be constructed. We construct a graph using the weighted Delaunay tetrahedrization of a spherical cover: the support centers for sites and the squared radii for weights. The vertices and edges of this weighted Delaunay tetrahedrization become nodes and edges of a graph. Our graph has additional two special nodes called as terminal nodes. Two terminal nodes generally referred as source and sink respectively represent conceptually the inside and outside of the object. Note that terminal nodes are purely symbolic and do not have corresponding vertices of the tetrahedral mesh. Edges incident to a terminal node is referred as terminal edges. In our graph, each of terminal nodes has terminal edges whose the other ends are each vertex of the tetrahedral mesh. The structure of the graph to be used in Graph-cut

is summarized as follows; graph has nodes which are composed of source, sink, and all the vertices of the weighted Delaunay tetrahedrization, and has edges which are consisted of terminal edges from each terminal nodes to each vertex of the tetrahedral mesh and all the edges of the tetrahedral mesh.

In our algorithm, vertices labeled as source represent the points inside the object, and the other vertices (labeled as sink) represent the outside. We set weak constraints so that vertices with the values of $f(\boldsymbol{v}) \geq 0$ tend to be classified into the source side and otherwise into the sink side.

We assign the following cost to a terminal edge whose one end is vertex v of the tetrahedral mesh:

$$\begin{cases} w(v, \text{source}) = 0, & w(v, \text{sink}) = k\frac{|f(\boldsymbol{v})|}{l_v} & (f(\boldsymbol{v}) < 0) \\ w(v, \text{sink}) = 0, & w(v, \text{source}) = k\frac{|f(\boldsymbol{v})|}{l_v} & (f(\boldsymbol{v}) \geq 0) \end{cases} \tag{5.1}$$

where l_v is the average length of edges incident to v. A constant k means a ratio of cost for terminal edges to cost for tetrahedral mesh edges. Fig. 5.8 summarizes these weighting. The left images are examples of weights assigned for terminal edges and normal edges and the right images shows a part of graph and corresponding scalar fields of $f(\boldsymbol{x})$. Performing Graph-cut on the graph in the upper right image classifies the vertices into outside and inside as shown in the lower right image in which edges belong to cut-set are not drawn. According to our experiments, we recommend to set k to about $5 \sim 15$. Details are discussed in the paragraph Parameter Setting.

The cost for a tetrahedral mesh edge with end points v_i and v_j is

$$w(v_i, v_j) = \frac{|f(\boldsymbol{v}_i) + f(\boldsymbol{v}_j)|}{l_{v_i, v_j}} \tag{5.2}$$

where l_{v_i, v_j} is the length of the edge. Since value $f(\boldsymbol{x})$ is an approximation of the signed distance of $\boldsymbol{x}$ from the surface, this cost increases for edges far from the surface. On the other hand, it decreases for edges crossing the surface because the signs of $f(\boldsymbol{x})$ at end points are different each other. Therefore crossing edges tend to belong to a cut-set. For the implementation of Graph-cut, we utilized codes distributed by Boykov and Kolmogorov based on [BK04].

If approximation with $f(\boldsymbol{x})$ is completely correct, the classification with respect to the sign of $f(\boldsymbol{x})$ should coincide with the classification in terms of Graph-cut. Actually inconsistent nodes where classifications are not consistent exist; such a point is indicated as the red-circled point in the image on the left of Fig. 5.7 and the right image of Fig. 5.8. As shown in Fig. 5.7, approximations have failed because of outliers. Graph-cut provides reliable classification even for such areas.

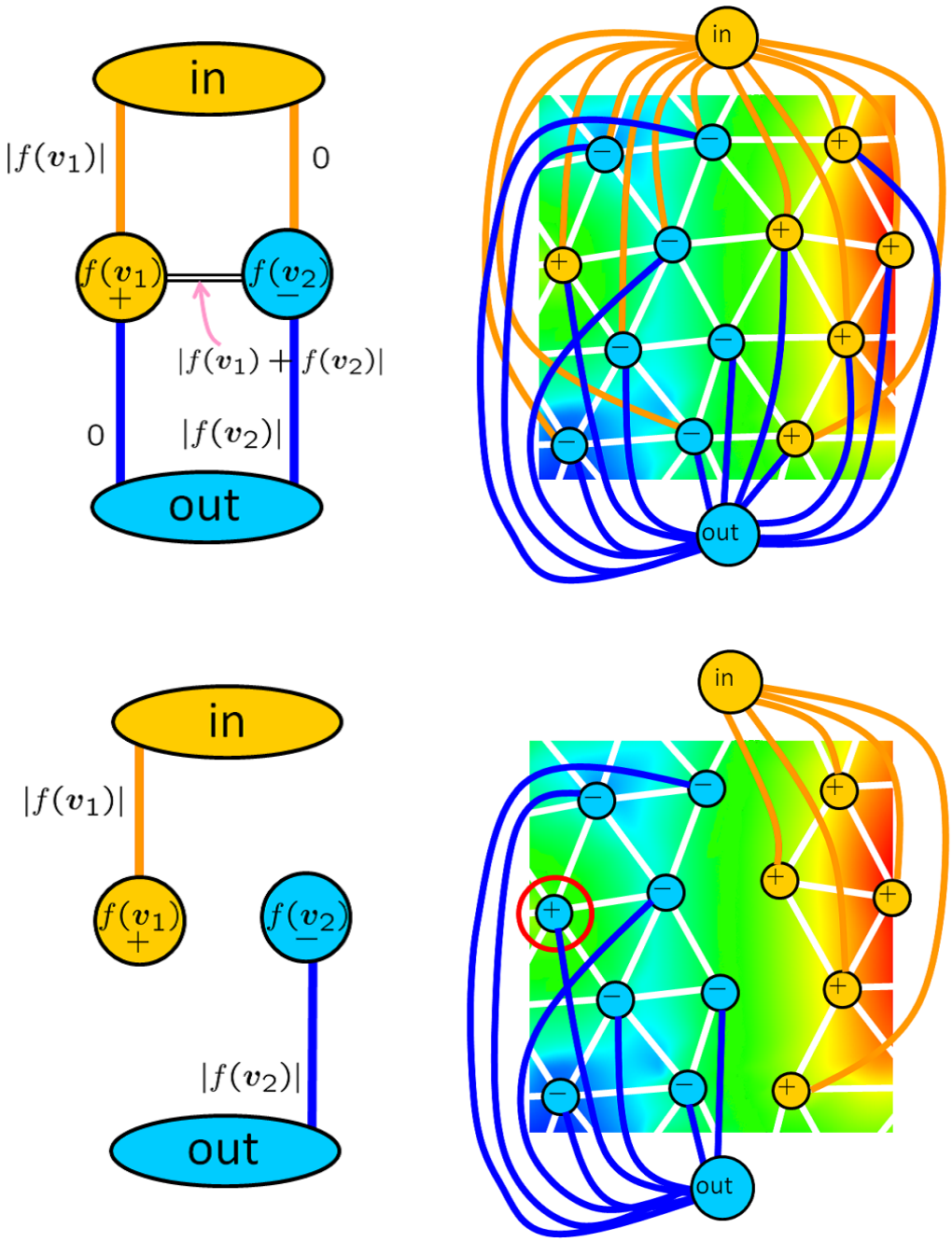


Fig. 5.8. Performance of weighting for Graph-cut.
Left: examples of weights assigned for terminal edges and normal edges. Right: a part of graph.
Upper: the initial graph. Lower: graph after Graph-cut. Cut-set is not drawn.

Triangular mesh extraction. In this chapter, our goal is integrating classification with Graph-cut and smoothing of local approximations (details are described in the next section). But it is sure that the reconstruction algorithm using Graph-cut is a strong method by itself. However, surface obtained in Step 2 cannot be detected with existing isosurface-

polygonization method as ones in Appendix C. So we give a polygonization method specialized for a tetrahedral mesh for a case that the diffusion technique will not be performed and surface is obtained only through combination of PU and Graph-cut.

The aim of this section is, performing the marching tetrahedra approach for extraction of a triangulated surface from a tetrahedral mesh [PT90, GH95]. Tetrahedra which have object-inside point(s) and object-outside point(s), the pink tetrahedra in Fig. 5.7, are considered to intersect with the surface. Here the inside/outside classification means classification result determined using Graph-cut. In this paragraph, edges consisting of cut-set are called intersecting edges, and points intersecting with the surface are describes as intersecting points.

In marching tetrahedra, for each tetrahedron, first intersecting edges are extracted then a polygonized surface is generated by connecting intersecting points detected on each intersecting edges. In the left image of Fig. 5.7, intersecting points are shown with green points and 2D mesh connecting intersecting points is drawn in a light green line. We determine an intersecting point as a point satisfying $f(\boldsymbol{x}) = 0$ on an intersecting edge. Fig. 5.9 shows the patterns of tetrahedra which intersect with the surfaces. Generated surfaces and intersecting points are colored with green. In general cases, intersecting points are obtained with regula falsi [PTVF93].

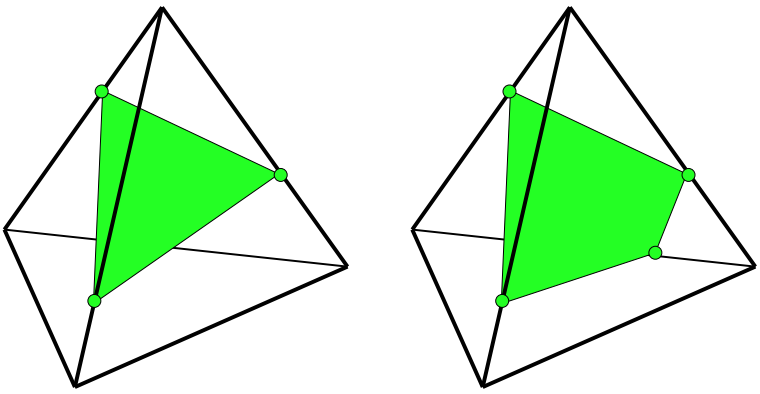


Fig. 5.9. Patterns of surfaces generated in tetrahedra.

In our work the signs of $f(\boldsymbol{x})$ may not be coincident with the classification with Graph-cut. See Fig. 5.10. Inner vertices are shown with yellow squares while outer vertices with blue circles. The signs inside vertices show the signs of $f(\boldsymbol{x})$. The left image shows an ordinary case. The ends of the nearest intersecting edge are consistent. In the right image,

the red-circled vertex with $f(\boldsymbol{x}) > 0$ is classified as outside by the result of Graph-cut. Hence the nearest edge is an intersecting edge which has the same signed endpoints. For such an intersecting edge, we temporally determine the coordinates of the intersecting point as the coordinates of one end point whose absolute value of $f(\boldsymbol{x})$ is less than the others.

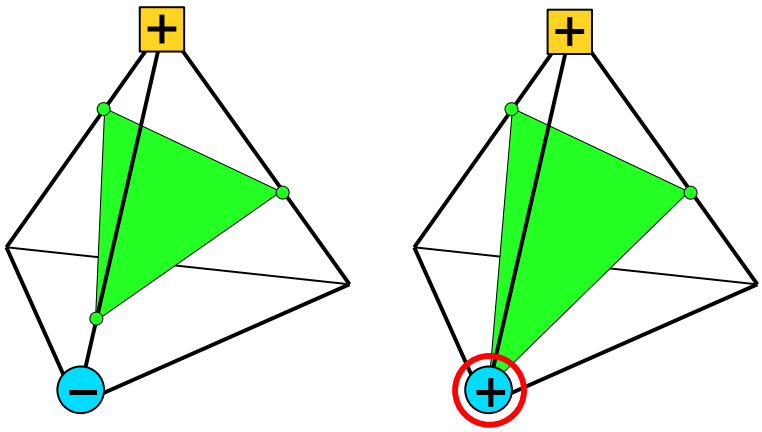


Fig. 5.10. Patterns of surfaces in consistent and inconsistent tetrahedra.
Left: surface in a tetrahedron with consistent edge. Right: surface in a tetrahedron with inconsistent edge.

Hereafter, the term vertex means vertex of a triangular mesh generated through the above modified marching tetrahedra. After triangular mesh generation, obtained surface mesh has some nonsmooth areas as shown in the left images of Fig. 5.11. This is because some vertices are temporally set on intersecting points without evaluation of appropriate positions. Since our goal is generation of smooth surface, we smooth the surface mesh with bi-Laplacian mesh smoothing. In addition to these areas, we also smooth under-sampled area which is another reason of nonsmoothness. In this section, under-sampled area is notified as a vertex which is referred to as under-sampling vertex. Remember that the vertex of the tetrahedral mesh is the center of a spherical support. We determine under-sampling vertex as vertex on a tetrahedral mesh edge which has at least one endpoint the corresponding support including less than ten sampling points.

For calculation of new position of vertex to be smoothed, we iterate the Laplacian smoothings in two directions [Tau95, KCVS98]. First a vertex $\boldsymbol{v}_i$ is moved to position $\boldsymbol{v}_i'$ following the equation

$$\boldsymbol{v}_i' = \boldsymbol{v}_i + \lambda \mathcal{U}, \tag{5.3}$$

then moved to a new place determined by the Laplacian smoothing in the reverse direction

$$\boldsymbol{v}_i'' = \boldsymbol{v}_i' - \lambda \mathcal{U}'. \tag{5.4}$$

where λ is a constant number which we set to 0.5, and $\mathcal{U}$ and $\mathcal{U}'$ are operators defined by

$$\mathcal{U} = \frac{1}{n}\sum_j \boldsymbol{v}_j - \boldsymbol{v}_i, \qquad \mathcal{U}' = \frac{1}{n}\sum_j \boldsymbol{v}_j' - \boldsymbol{v}_i' \tag{5.5}$$

where $\boldsymbol{v}_j$ is a vertex incident to $\boldsymbol{v}_i$ and n is the degree of vertex $\boldsymbol{v}_i$.

Note that this smoothing is performed only for vertices specified above, and no other vertices are moved. The results in this section are obtained with 30 times iterations. With these settings, the movement of vertices almost converges.

Fig. 5.11 shows the effects of bi-Laplacian smoothing.

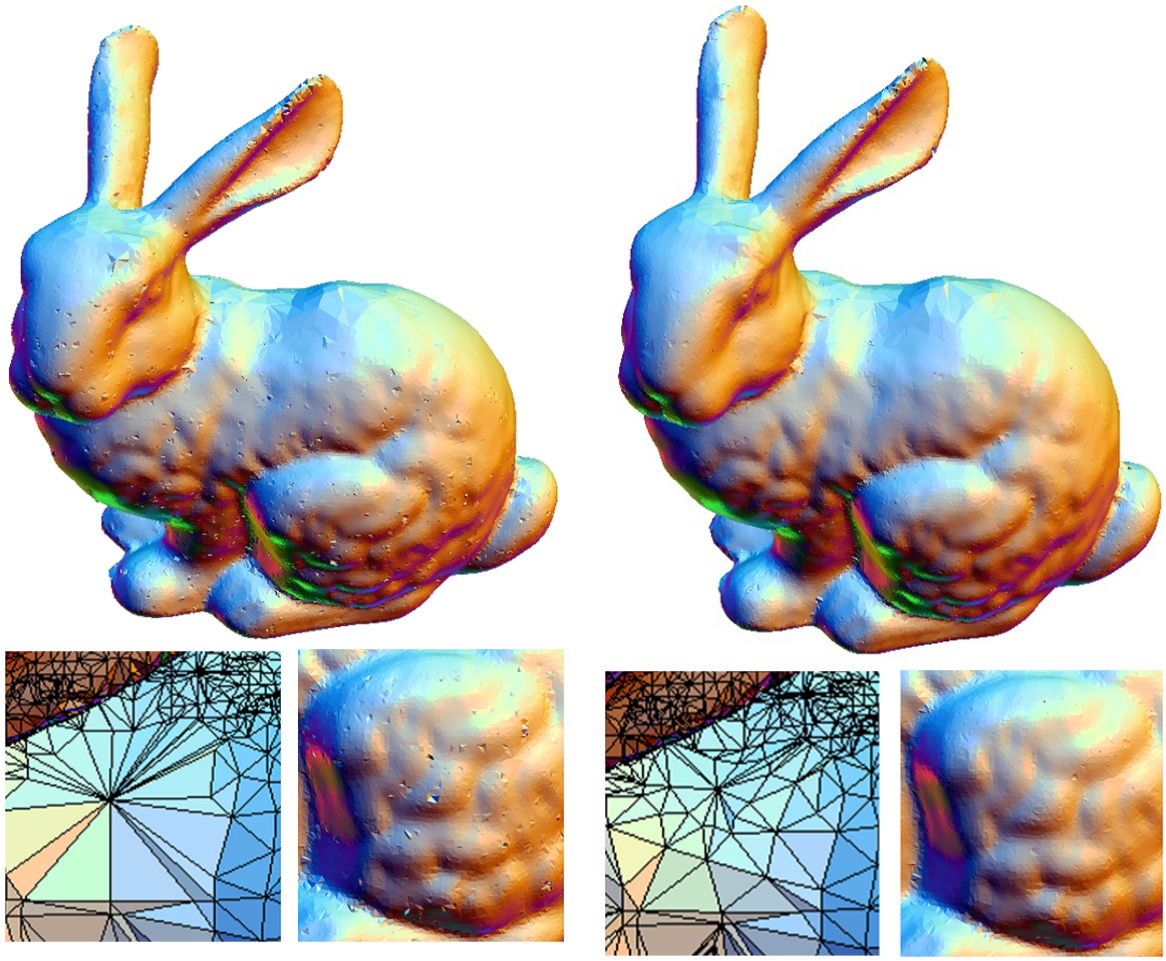


Fig. 5.11. Results of bi-Laplacian smoothing. Lower: close-ups of the backs (wireframe) and the left hind feet.
Left: surfaces before bi-Laplacian smoothing. Right: surfaces after bi-Laplacian smoothing.

Algorithm Summary

Here we summarize the surface reconstruction algorithm using combination of partition of unity implicits and Graph-cut.

Algorithm: surface reconstruction using PU and Graph-cut

1. Generate a spherical cover for the bounding box through partition of unity
2. Construct a graph with the aid of weighted Delaunay tetrahedrization and perform Graph-cut
3. Polygonize the surface

Step 1 is performed as explained in Section 3.4.1. The details of Step 2 are described in this section. If obtained surface have to be polygonized, modified marching tetrahedra approach is performed in Step 3. The goal of this chapter is reconstruction performing Graph-cut and smoothing (descried in the next section) however, the above summary is for reconstruction only with Graph-cut approach, without smoothing.

Experimental Results

Parameter setting. Results with different values of k in equation (5.1) are shown in Fig. 5.12. The reconstructed shapes are affected by the values of k. In the result with $k = 3$, shown in the center of Fig. 5.12, the lower jaw and the end of the antler are not be reconstructed. On the other hand, in the result with $k = 15$, the right image of Fig. 5.12, these parts are completely appear. In our experiments, setting a larger value for k, some parts that do not appear with smaller values can be constructed.

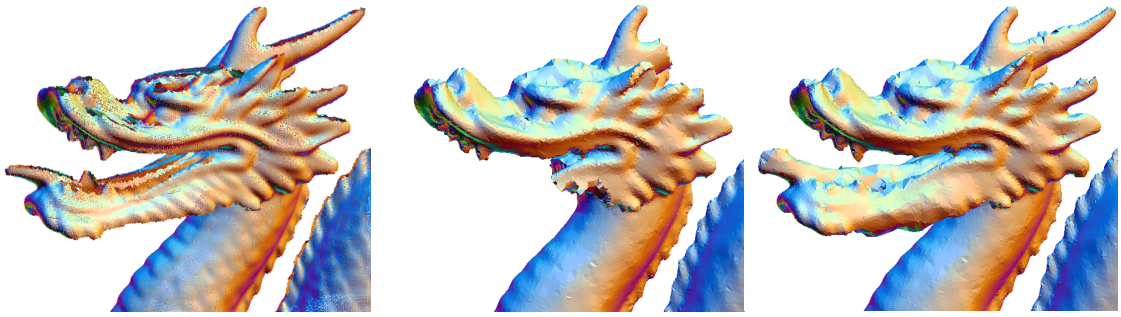


Fig. 5.12. Reconstructed models with different k values in equation (5.1).
Left: sampling points.
Center: result with $k = 3$.
Right: result with $k = 15$.

Robustness. In Fig. 5.13, we show the results for the Stanford dragon raw data that is very noisy and has areas of undersampling as shown in the close-up in the left image. In the result meshes in the center and the right images, holes are filled with desired-shape meshes. These results exemplify the advantage of a global method to handle data with lack of sampling.

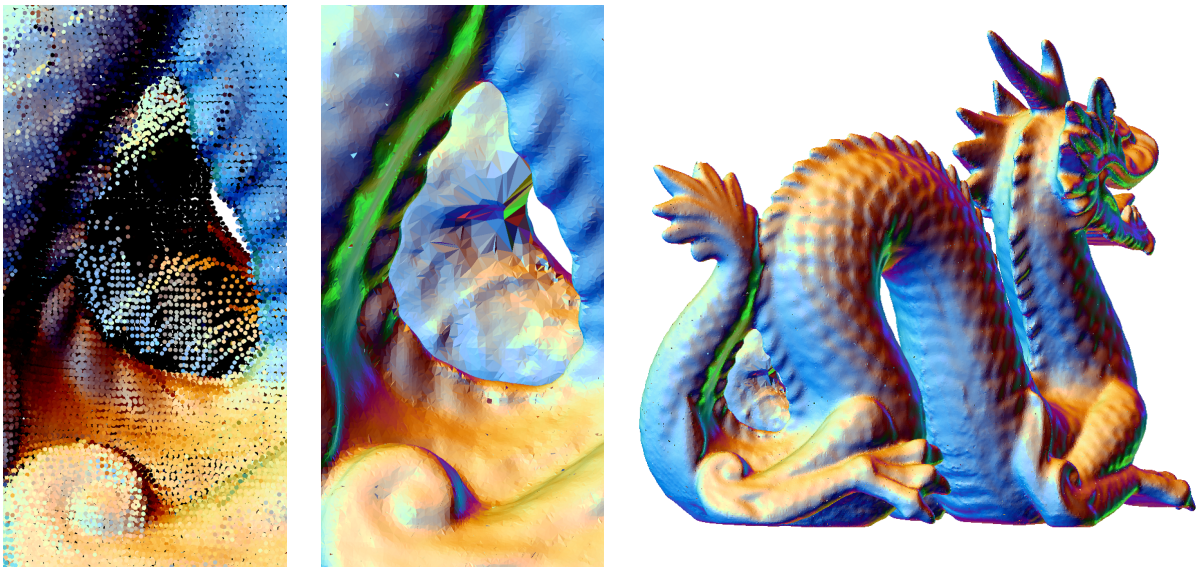


Fig. 5.13. Hole-filling effect achieved with the aid of introducing Graph-cut.
Left: close-up of a hole caused by lack of sampling.
Middle: close-up of a hole-filled mesh.
Right: whole of the Stanford dragon.

Performance. RAM is required about 500 Mbytes for processing one million input points, and it increases linearly.

For processing a few million points, proposed algorithm takes a few tens of minutes. This is a issue to be solved and we discuss about this problem in Section 5.3.

Comparison. Comparison for data with several outliers and lack of sampling, is shown in Fig. 5.14. The sampling points are shown in (a). Parameters are set for the error tolerance to 1.0×10^{-3} in MPU, maximum depth of octree for 9 in Poisson surface reconstruction, error tolerance ε to 1.0×10^{-3} in ours. The result obtained using MPU, (b), has largely deformed shapes on its back and top of head which are considered as influences of lack of sampling. In addition, some extracomponents appears near the tail. Poisson surface reconstruction gives a roughly desired shape,(c), but its smoothing effect for noise included in this data is too strong and the details in eyes and body are smoothed out. The combination of local and global algorithm gives a smooth mesh with precise representation without being affected by outliers and lack of sampling, see the image in (d).

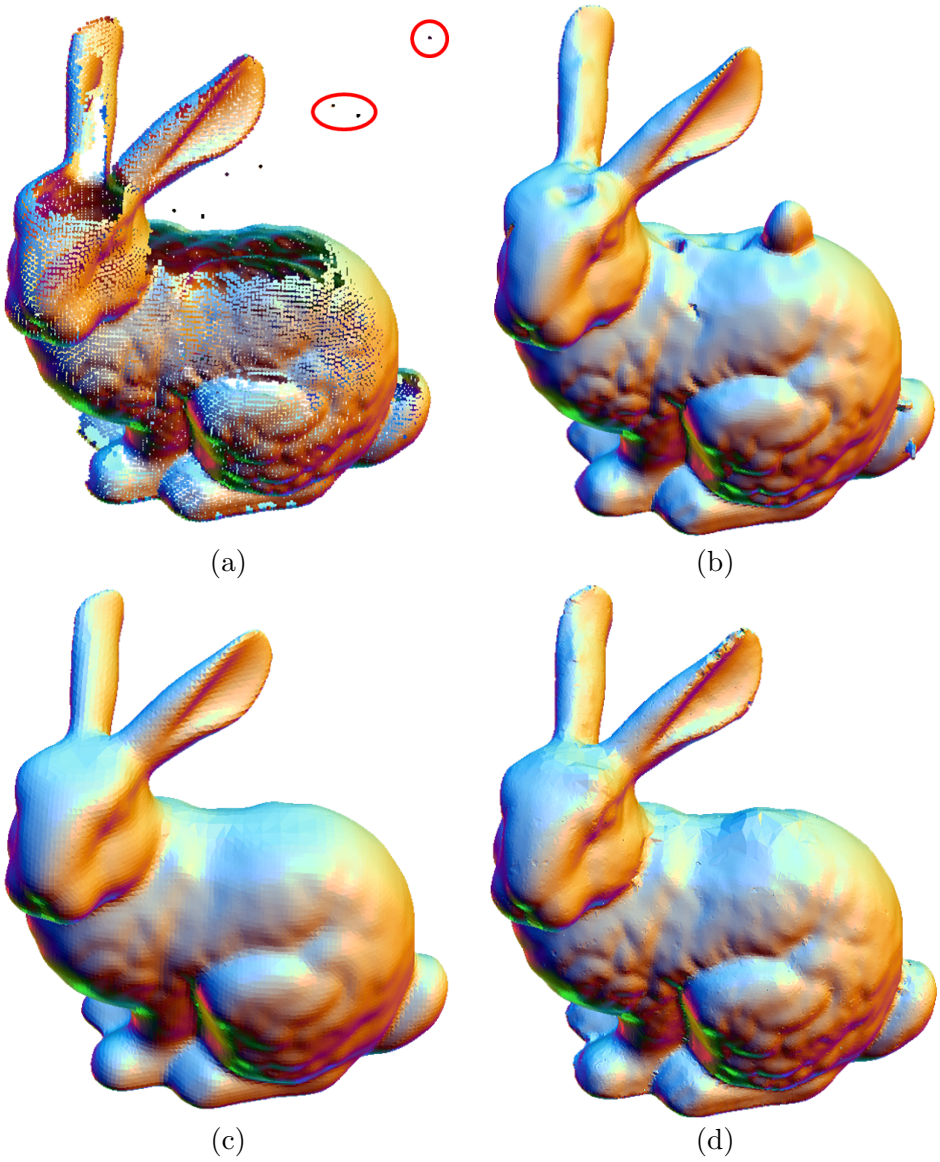


Fig. 5.14. Comparison of surface reconstruction through Graph-cut approach for Stanford bunny with other methods.
(a) Sampling points involving outliers (red-circled points) and lack of sampling.
(b) Result generated using MPU.
(c) Result generated Poisson surface reconstruction.
(d) Result generated combination of partition of unity and Graph-cut.

Here are another examples showing the effects of combining Graph-cut. In Fig. 5.15, we compared our results with MPU proposed by Ohtake et al. [OBA+03] and with Poisson surface reconstruction proposed by Kazhdan et al. [KBH06]. Input data is low-quality scanned data ((a)). Especially challenging parts are fangs in the mouth and details of the fingers. As parameters, for Poisson surface reconstruction we set the maximum octree level as 10, for MPU, error tolerance is set as 1.0×10^{-3}, and for ours, the octree level as

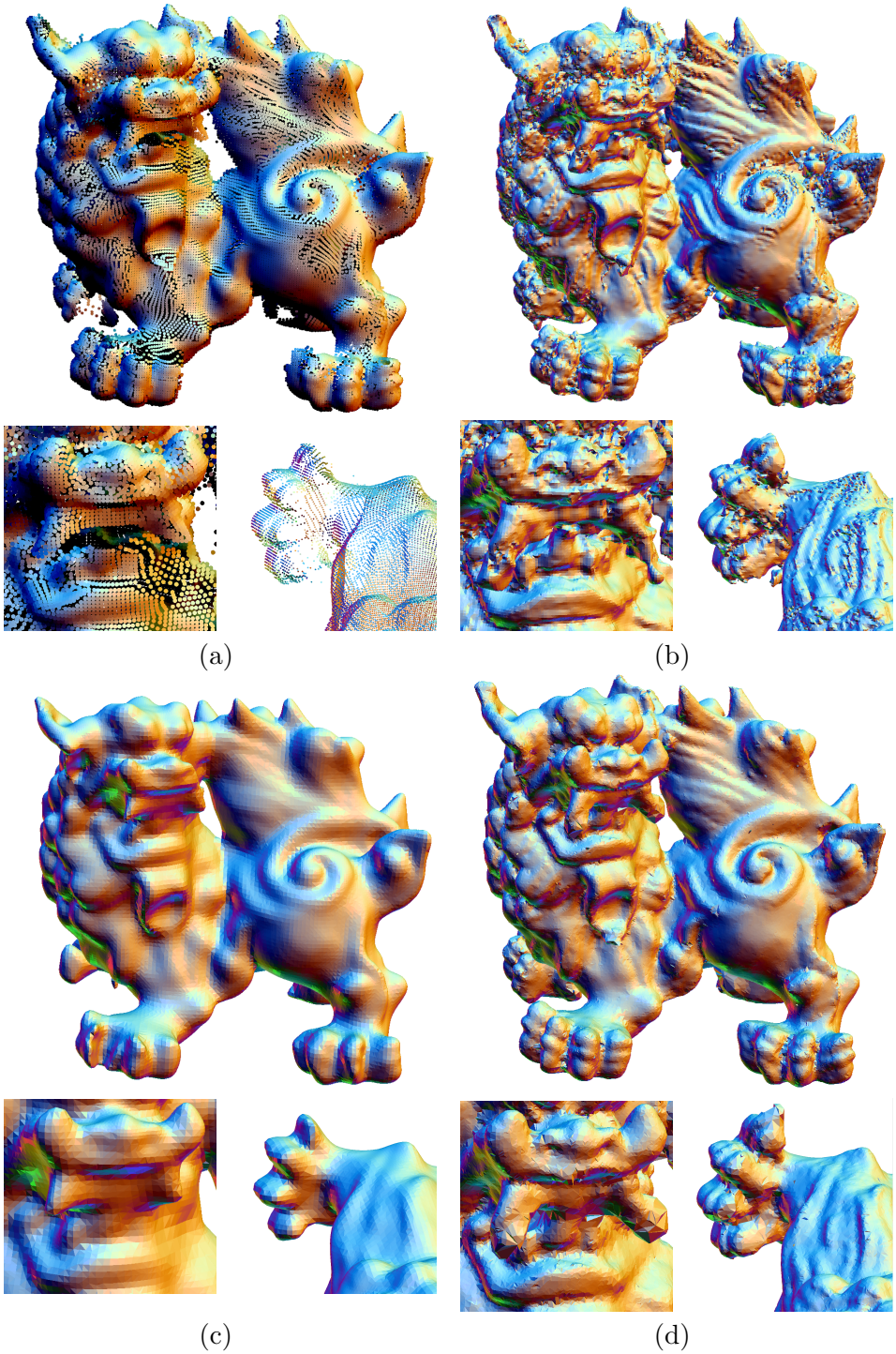


Fig. 5.15. Comparison of surface reconstruction through Graph-cut approach for Shiisa-A with other methods. In the lower columns, close-up views of the mouths and right feet are shown.
(a) Sampling points.
(b) Results generated using MPU.
(c) Results generated using Poisson surface reconstruction.
(d) Results generated using proposed algorithm.

8 and the error tolerance ε as 1.0×10^{-3}.

Fig. 5.15(b) shows surface generated through MPU which have many tiny extra components. Note that MPU approximates objects with quadratic functions while proposed algorithm uses linear functions. It enables MPU generate very precise results, but fitting the second order approximations is sensitive to noise.

Another reconstructed surface generated with Poisson surface reconstruction which is a global approach is in Fig. 5.15(c). Poisson surface reconstruction can generate watertight meshes, but the results may end to be too smooth and many important details may be lost. Moreover, especially for sparse and noisy data, the results tend to be shrunk, see the close-up of the foot.

Our algorithm can deals with low-quality scanned data as Poisson surface reconstruction, see Fig. 5.15(d). Moreover fangs and details of the fingers are successfully reconstructed. The reason is that Poisson surface reconstruction is the zeroth order approximation (a scalar value is assigned to each octant), while ours is the first order approximation.

5.2.3 Diffusing Local Approximations

Introduction

For smooth shape representation, Laplacian smoothing of a gradient field is performed on a spherical cover. Since we operate geometrical processings on a spherical cover, using discrete differential operators specialized for a set of spherical supports is preferred than using general differential operators for a traditional orthogonal grid. Not only smoothing a gradient field, but also the local approximations are smoothed reflecting the smoothed gradient field.

Our smoothing for gradients of local approximation functions is based on the diffusion of a gradient field. Fig. 5.16 represents the main idea. Isosurfaces of local and global approximations are shown with dotted and solid lines respectively. In the initial state (the left image), each local approximations are not matched with surrounding local approximations. So the obtained global isosurface in this state is not smooth. Proposed diffusion technique makes local approximations consistent with each other (the right image). As a result, a smooth global isosurface is obtained.

This smoothing can handle adaptively sampled data, and enables topological change unlike the existing smoothing techniques. Experimental results for 3D sampling points are shown in Fig. 5.17 and 5.18. In both examples, in initial surfaces, strong influences of noise appear as bumps and extracomponents, but do not exist in the surface after smoothing. Disappearance of extracomponents means a topological contribution of proposed algorithm.

The concept is inspired by normal-based smoothing methods [Tau01, OBS02, CT03] that smooth normals then move vertices. The novelty of our approach lies in its use of differential operators on a spherical cover with linear functions.

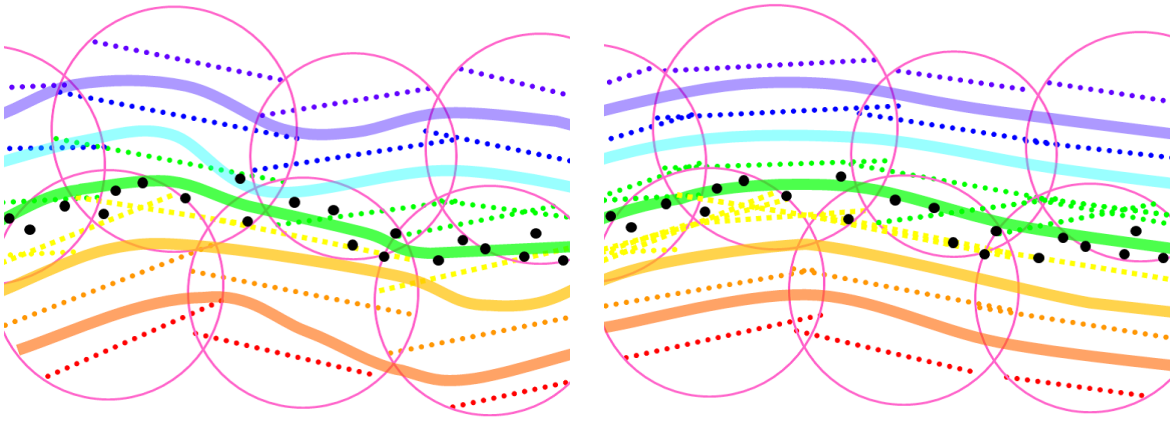
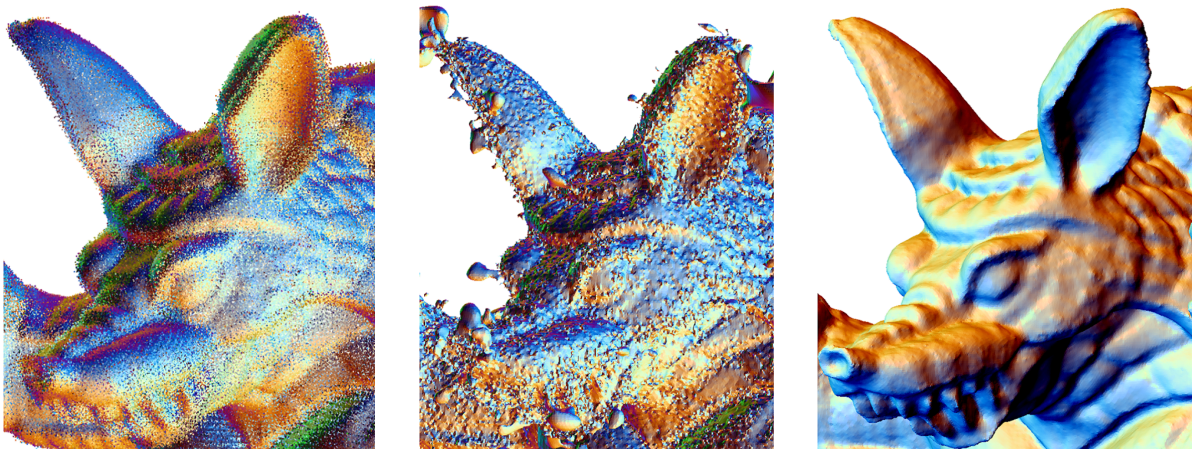


Fig. 5.16. 2D images of diffusion of local approximation functions.
Left: isosurfaces generated using PU.
Right: isosurfaces after the smoothing of local approximation functions.

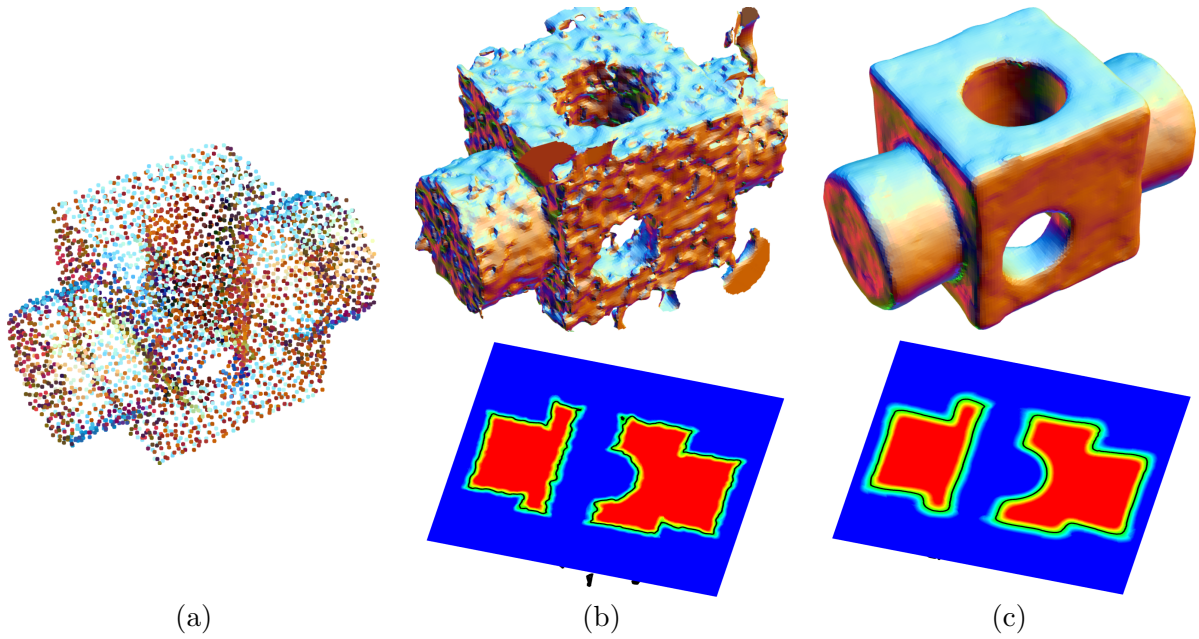


(a) (b) (c)

Fig. 5.17. Examples of surface reconstruction with diffusion.
(a) Sampling points. (b) Result generated using only partition of unity approach. (c) Result generated using partition of unity approach and diffusion technique.

Algorithm

After local approximation functions are obtained through spherical covering, smoothing of the gradient field and updating of local approximations are alternately iterated. In each iteration cycle, first, gradients of $f(\boldsymbol{x})$ is smoothed through Laplacian smoothing with the



(a) (b) (c)

Fig. 5.18. Effects of the smoothing of local approximation functions.

(a): Sampling points. (b): Initial surface obtained using PU. (c): Results of smoothing for the model in the left image. For (b) and (c), in upper row, the whole objects and in the lower row, cross-sections are displayed.

discrete Laplacian operator on a spherical cover. Then, each local approximation function is updated to have the smoothed vector as its gradient.

Here we show an overview of one cycle of iterative PU smoothing. Fig. 5.19 shows the overview of the whole smoothing.

Algorithm: smoothing partition of unity implicits

1. Initialize the vector field with the gradient of $f(\boldsymbol{x})$:
   set $\boldsymbol{v}_i \equiv \nabla f(\boldsymbol{c}_i)$ for each sphere center $\boldsymbol{c}_i$
2. Update $\{\boldsymbol{\alpha}_i\}$: smooth $\boldsymbol{v}_i$ with Laplacian smoothing. Set updated vector $\hat{\boldsymbol{v}}_i$ to $\hat{\boldsymbol{\alpha}}_i$
3. Update $\{\beta_i\}$: calculate $\hat{\beta}_i$ satisfying $\hat{\boldsymbol{v}}_i = \mathrm{Grad}\,(f(\boldsymbol{c}_i))$. Update β_i to $\hat{\beta}_i$

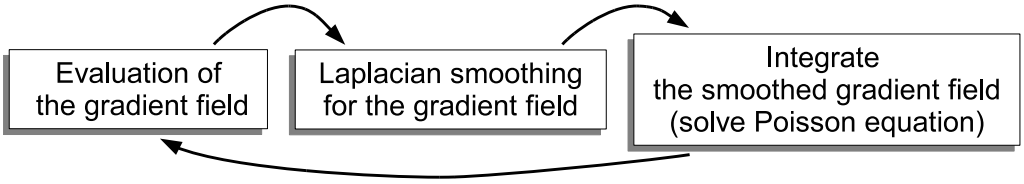


Fig. 5.19. Overview of smoothing of partition of unity implicits.

The details of each steps are described in the following sections.

Generating Gradient Field

The aim of the first step of the **Algorithm: smoothing partition of unity implicits** is, initializing a gradient field with the gradient of an implicit function $f(\boldsymbol{x})$. Remember that function $f(\boldsymbol{x})$ is determined by the weighted average of local approximations:

$$f(\boldsymbol{x}) = \frac{\sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x})}{\sum_i w_i(\boldsymbol{x})}. \tag{5.6}$$

where $g_i(\boldsymbol{x})$ is a local approximation function and $w_i(\boldsymbol{x})$ is a weighting function. The gradient of $f(\boldsymbol{x})$ is

$$\begin{aligned}\nabla f(\boldsymbol{x}) = &\left(\nabla\left(\sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x})\right)\sum_i w_i(\boldsymbol{x}) - \sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x}) \nabla\left(\sum_i w_i(\boldsymbol{x})\right)\right) \\ &\Big/\left(\sum_i w_i(\boldsymbol{x})\right)^2 \\ = &\left(\left(\sum_i \nabla w_i(\boldsymbol{x}) g_i(\boldsymbol{x}) + \sum_i w_i(\boldsymbol{x}) \nabla g_i(\boldsymbol{x})\right)\sum_i w_i(\boldsymbol{x})\right. \\ &\left. - \sum_i w_i(\boldsymbol{x}) g_i(\boldsymbol{x})\left(\sum_i \nabla w_i(\boldsymbol{x})\right)\right) \\ &\Big/\left(\sum_i w_i(\boldsymbol{x})\right)^2 .\end{aligned} \tag{5.7}$$

Since $f(\boldsymbol{x})$ is a linear combination of linear functions $\{g_i(\boldsymbol{x})\}$, its gradient $\nabla f(\boldsymbol{x})$ is a constant vector.

In Step 1, each support center $\boldsymbol{c}_i$ is assigned the value of above gradient $\nabla f(\boldsymbol{c}_i)$ as its initial vector $\boldsymbol{v}_i$. Note that this initialization satisfies Assumption 4.3.1 in Chapter 4.

Smoothing of Gradient Field

In Step 2, vectors $\{\boldsymbol{v}_i\}$ are smoothed with Laplacian smoothing. Laplacian smoothing replaces a vector $\boldsymbol{v}_i$ with a new vector $\hat{\boldsymbol{v}}_i$ satisfying

$$\mathrm{Lap}(\hat{\boldsymbol{v}}_i) = 0. \tag{5.8}$$

In the above equation, Lap means the discrete Laplacian operator on a spherical cover, defined in equation (4.27) in the previous chapter. Following equation (4.27) and (5.8), smoothed vector $\hat{\boldsymbol{v}}_i$ can be obtained through solving

$$\begin{aligned}\mathrm{Lap}(\hat{\boldsymbol{v}}_i) &= \frac{3}{r_i S_i}\sum_j \phi_{i,j}(\boldsymbol{v}_j - \hat{\boldsymbol{v}}_i) \\ &= 0.\end{aligned} \tag{5.9}$$

From above, equation

$$\hat{\boldsymbol{v}}_i \sum\nolimits_j \phi_{i,j} = \sum\nolimits_j \phi_{i,j} \boldsymbol{v}_j \tag{5.10}$$

is derived. Now smoothed vector is obtained:

$$\hat{\boldsymbol{v}}_i = \frac{\sum_j \phi_{i,j} \boldsymbol{v}_j}{\sum_j \phi_{i,j}}. \tag{5.11}$$

Update of Local Approximation Functions

In the previous paragraph, how to smooth the gradient field is described. In the final step, local approximation functions are updated reflecting the smoothing of the gradient field.

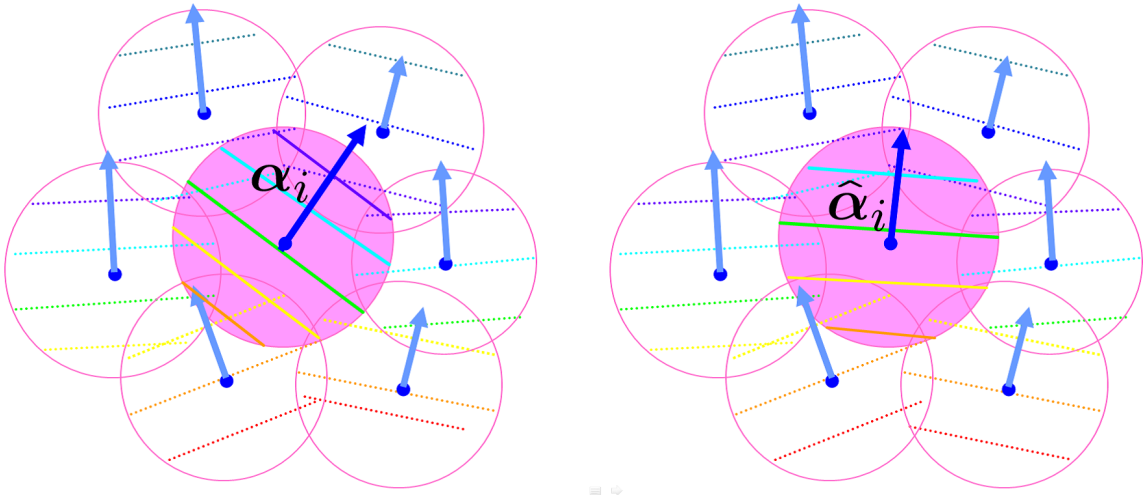


Fig. 5.20. Updating of a coefficient vector of a local approximation function in 2D. Spherical supports with coefficient vectors and isosurfaces of associated local approximations.
Left: initial state.
Right: after updating of a coefficient vector of the pink-colored support.

We update local approximations $\{g(\boldsymbol{x})\}$ with Poisson equation reflecting the smoothing of gradients $\{\boldsymbol{v}_i\}$. This updating is consisted of two steps: updating coefficient vectors $\{\alpha_\mathbf{i}\}$ and updating constant terms $\{\beta_i\}$.

The update of a coefficient vector $\boldsymbol{\alpha}_i$ to $\hat{\boldsymbol{\alpha}}_i$ is realized by a substitution

$$\hat{\boldsymbol{\alpha}}_i = \hat{\boldsymbol{v}}_i. \tag{5.12}$$

An updated coefficient vector $\hat{\boldsymbol{\alpha}}_i$ is a weighted average of gradient vectors of neighbors of s_i, a support under consideration, see equation (5.11). Fig. 5.20 shows the effects of updating of a coefficient vector. The left image shows the initial state. The blue-colored arrow is a coefficient vector $\boldsymbol{\alpha}_i$ of the pink-colored support, it is also the gradient vector because $g_i(\boldsymbol{x})$ is linear. $\boldsymbol{\alpha}_i$ is not matched with surrounding gradients. After updating, a

smoother vector field is obtained and the gradient of isosurfaces are matched with others, see the right image.

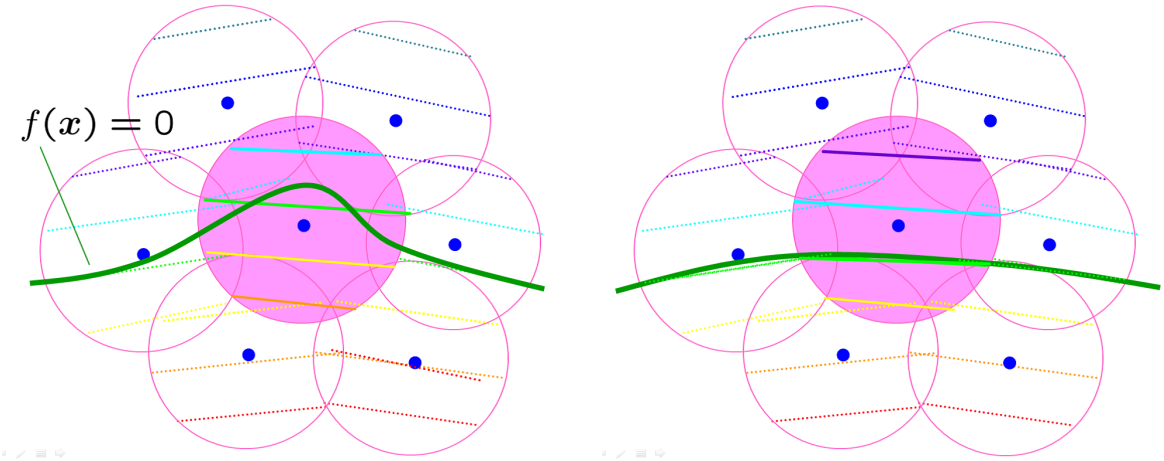


Fig. 5.21. Updating of a constant term of a local approximation function in 2D. Spherical supports with isosurfaces of local approximations, and a global isosurface.
Left: initial state, coefficient vectors have been smoothed.
Right: after updating of a constant term of the pink-colored support.

Updating constant terms is equivalent to updating $f(\boldsymbol{x})$ so that it satisfies

$$\mathrm{Grad}\,(f(\boldsymbol{c}_i)) = \hat{\boldsymbol{v}}_i. \tag{5.13}$$

For simple computation, here we make an assumption that the center of a spherical support is not included in other spherical supports, like stats in Fig. 5.20;

$$f(\boldsymbol{c}_i) \approx g_i(\boldsymbol{c}_i) = \beta_i. \tag{5.14}$$

We obtained good results for all examples in this chapter under this assumption.

As mentioned above, the purpose of this step is updating a scalar field satisfying equation (5.13). The left hand side of equation (5.13) is only composed of a divergence term, it implies that the smoothed vector field must be represented with no curl-term. This constraint is very strong, and this is the reason of that solving this equation directly is difficult. So we apply the discrete divergence operator on the both hand sides and solve the obtained weak form. Note that we can use the discrete divergence operator because the both hand sides of equation (5.13) are constant vectors. The obtained weak form is a Poisson equation:

$$\mathrm{Lap}\,(f(\boldsymbol{c}_i)) = \mathrm{Div}\,(\hat{\boldsymbol{v}}_i)\,. \tag{5.15}$$

In the following, how to solve equation (5.15) is explained. The solution is obtained solving a minimization problem:

$$\|\mathrm{Lap}\,(f(\boldsymbol{c}_i)) - \mathrm{Div}\,(\hat{\boldsymbol{v}}_i)\,\|^2 \to \min\,. \tag{5.16}$$

$\mathrm{Lap}\,(f(\boldsymbol{c}_i)) - \mathrm{Div}\,(\hat{\boldsymbol{v}}_i)$ is a linear function of β_i, a verification with expansion of this equation will be described later.

Before calculating a complex formula, let us consider a simplified linear function $ax + b$ for $x \in \mathbb{R}$. In this case, a minimization problem becomes

$$(ax + b)^2 \to \min\,. \tag{5.17}$$

It is clear that the minimizer x satisfies the below equation derived from differentiation with respect to x:

$$a(ax + b) = 0. \tag{5.18}$$

So the minimizer x is described as

$$x = -\frac{b}{a}. \tag{5.19}$$

Get back to the solution of a constant term β_i. First, let us verify that $\mathrm{Lap}\,(f(\boldsymbol{c}_i)) - \mathrm{Div}\,(\hat{\boldsymbol{v}}_i)$ is a linear function with regard to β_i:

$$\begin{aligned}
\mathrm{Lap}\,(f(\boldsymbol{c}_i)) - \mathrm{Div}\,(\hat{\boldsymbol{v}}_i) &= \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j}(f(\boldsymbol{c}_j) - f(\boldsymbol{c}_i)) \\
&\qquad - \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j} \hat{\boldsymbol{v}}_j \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i) \\
&= \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j}(\beta_j - \beta_i) \\
&\qquad - \frac{3}{r_i S_i} \sum\nolimits_j \phi_{i,j} \hat{\boldsymbol{v}}_j \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i) \\
&= -\frac{3}{r_i S_i} \sum_j \phi_{i,j} \beta_i + \frac{3}{r_i S_i} \sum_j \phi_{i,j} \left(\beta_j - \hat{\boldsymbol{v}}_j \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i)\right).
\end{aligned} \tag{5.20}$$

Therefore, the analytical solution of a linear function in equation (5.19) can be applied. In this case, the a and b of equation (5.17) are

$$a = -\frac{3}{r_i S_i} \sum_j \phi_{i,j} \tag{5.21}$$

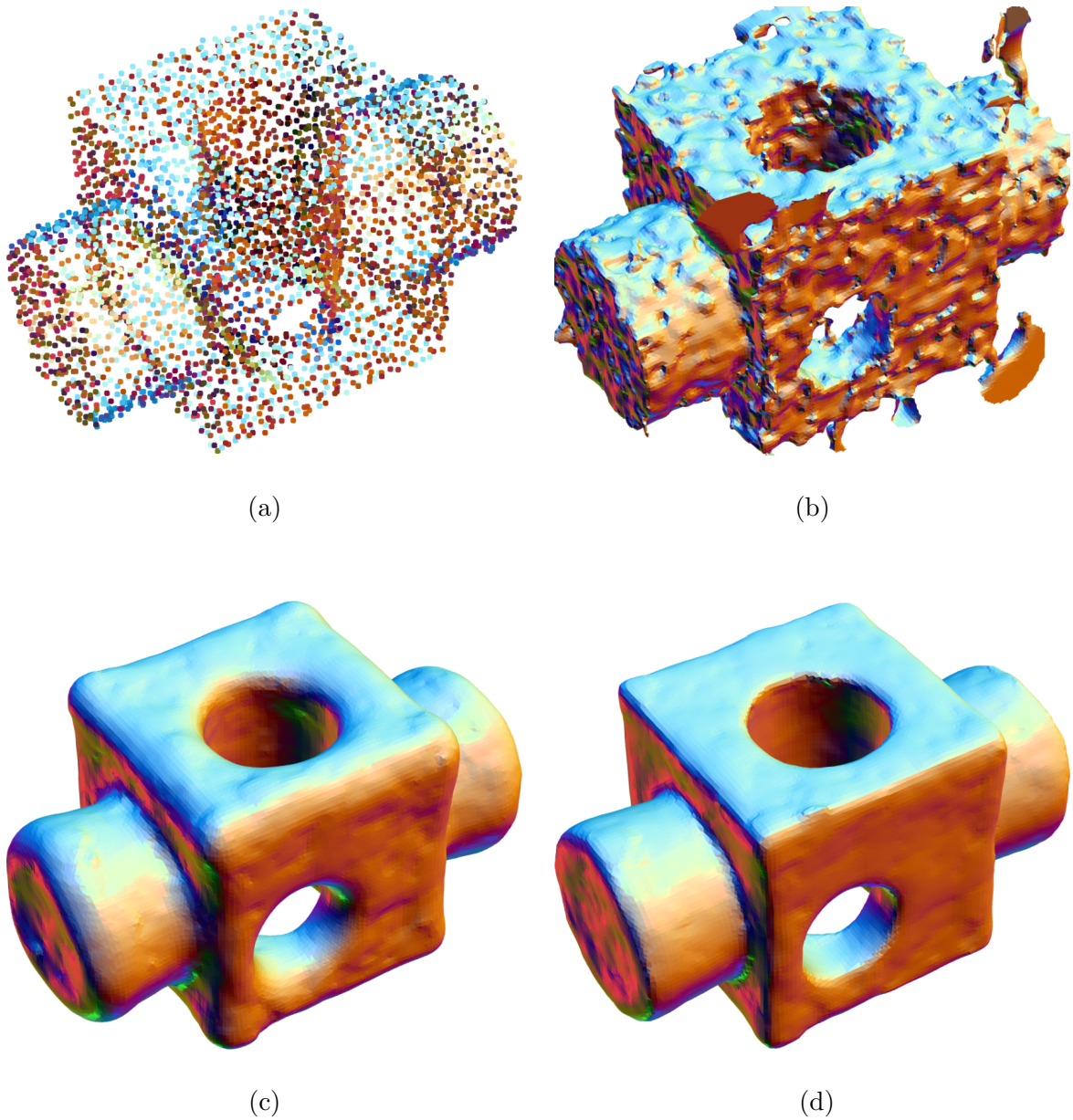
and

$$b = \frac{3}{r_i S_i} \sum_j \phi_{i,j} \left(\beta_j - \hat{\boldsymbol{v}}_j \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i)\right) \tag{5.22}$$

respectively. Therefore, the solution $\hat{\beta}_i$ to which β_i is updated becomes

$$\begin{aligned}
\hat{\beta}_i &= \frac{\frac{3}{r_i S_i} \sum_j \phi_{i,j} \left(\beta_j - \hat{\boldsymbol{v}}_j \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i)\right)}{\frac{3}{r_i S_i} \sum_j \phi_{i,j}} \\
&= \frac{\sum_j \phi_{i,j} \left(\hat{\boldsymbol{\alpha}}_j \cdot (\boldsymbol{c}_i - \boldsymbol{c}_j) + \beta_j\right)}{\sum_j \phi_{i,j}}.
\end{aligned} \tag{5.23}$$

Fig. 5.22 shows the effects of the above smoothing. Sampling points displayed in (a) includes some amount of noise. Surface directly reconstructed with partition of unity is shown in (b). Surfaces are very irregular affected by noise. After the smoothing of local approximation functions, these are smoothed out as shown in (c).



(a) (b)

(c) (d)

Fig. 5.22. Isotropic and anisotropic smoothing for the block model.
(a) Input points.
(b) Result of reconstruction with partition of unity approach.
(c) Result generated through isotropic smoothing.
(d) Result generated through anisotropic smoothing.

Anisotropic Smoothing of Gradient Field

As can be seen in Fig. 5.22 (c), models obtained through above smoothing may lose their edges. This detail lost arises because of the isotropy of the discrete differential operators proposed in Chapter 4. In order to preserve edges, we modify this smoothing scheme to anisotropic one introducing an anisotropic term into the smoothing in updating of $\boldsymbol{\alpha}_i$ and

β_i of local approximation functions.

To preserve highly curved regions, we propose the anisotropy factor similar to that proposed by Perona et al. [PSM94]. We propose the use of the angle-based anisotropic factor

$$\psi_{i,j} = \frac{1}{1+\theta_{i,j}^2}, \tag{5.24}$$

where $\theta_{i,j}$ is the angle between gradients $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. See Fig. 5.23 which shows the meaning of this anisotropic factor in 2D. This factor is introduced into the smoothing of $\boldsymbol{v}$ (equation (5.11)) and the updates of constant term of local approximation (equation (5.23)). The weighting function $\phi_{i,j}$ in these equations are replaced with $\phi_{i,j}\psi_{i,j}$.

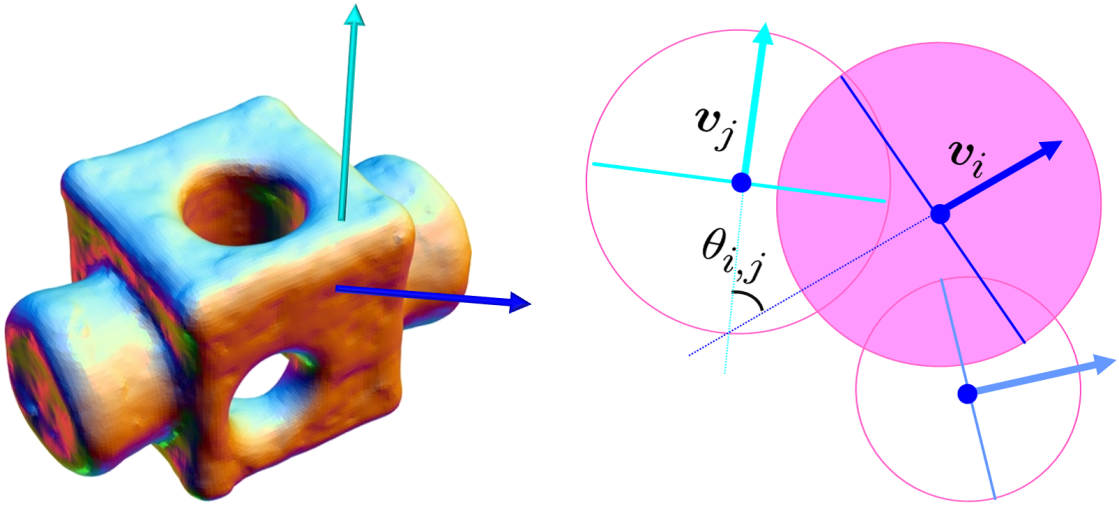
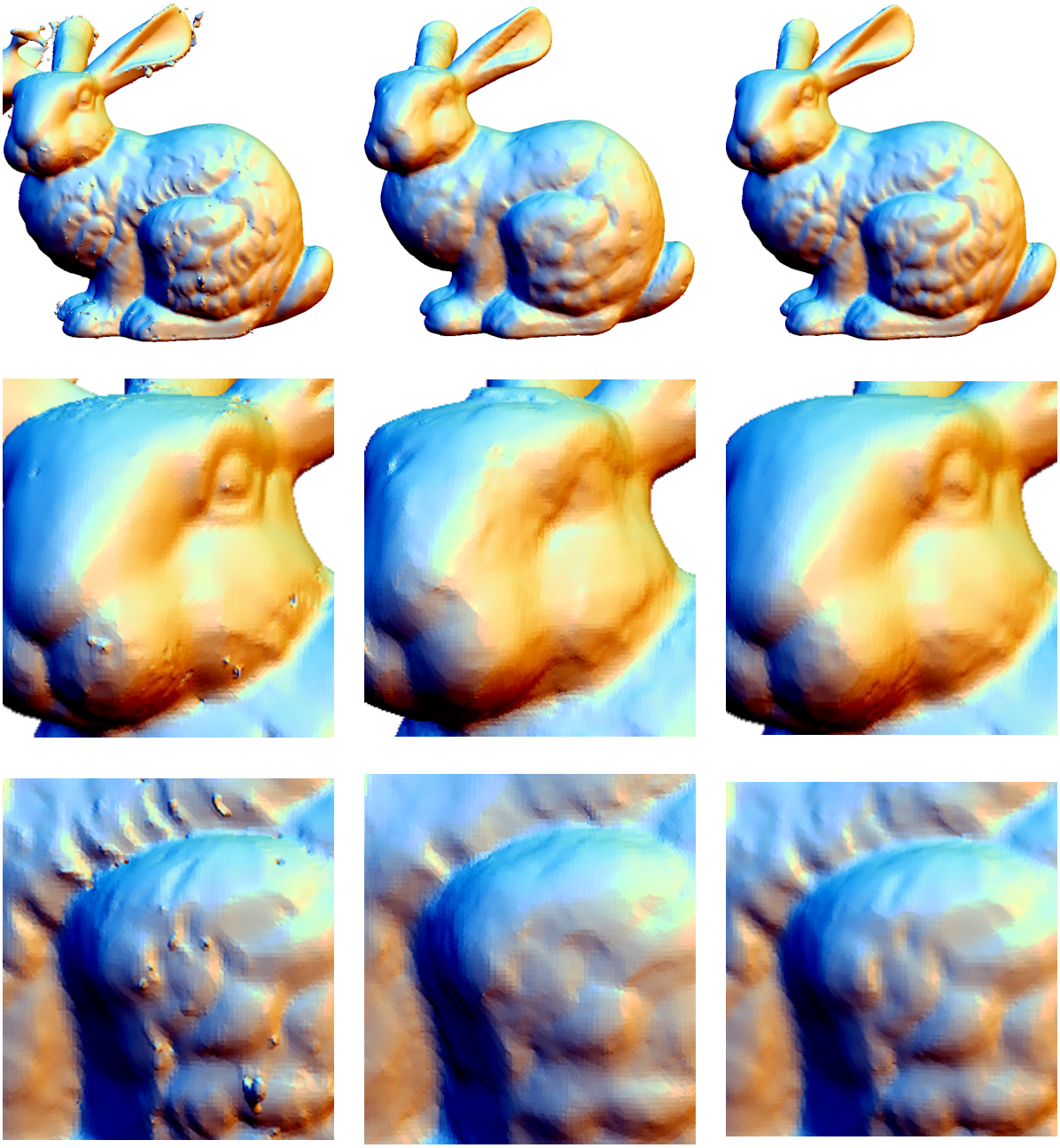


Fig. 5.23. Angle-based anisotropic factor.
Left: cross-edge gradients $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ with distinct directions.
Right: angle $\theta_{i,j}$ is made by $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$.

An example of the effect of this anisotropic smoothing is shown in Fig. 5.22 (d). With the aid of the anisotropic smoothing, sharp edges and corners are better preserved as shown in (d) than the result obtained with the isotropic smoothing in (c).

Detail Preservation

In surface reconstruction, preserving details is just as important as smooth representation. To avoid losing details caused by oversmoothing, we introduce terms that make the approximated surface fit the sampling points. The effects of this modification can be seen in Fig. 5.24. After applying several iterations of PU smoothing for an initial model, Fig. 5.24 (a), if the fitting terms are not applied, the details are smoothed out as shown by the image in (b). On the other hand, the results with the fitting terms demonstrate that the details are preserved well, see the image in (c).



(a) (b) (c)

Fig. 5.24. Detail preservation.
(a) Initial model generated with partition of unity approach.
(b) Smoothed model without fitting to sampling points.
(c) Smoothed model with fitting to sampling points.

- Detail Preservation for Coefficient Vectors

For smoothing and fitting a coefficient vector $\boldsymbol{\alpha}_i$, a term that makes $\boldsymbol{\alpha}_i$ fit to the

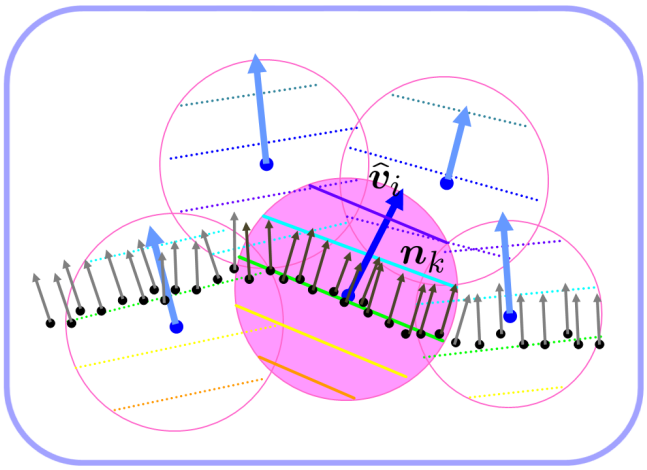


Fig. 5.25. Idea of detail preservation for updating of coefficient vectors.

weighted sum of the normals of sampling points is introduced. Let us consider a vector $\boldsymbol{v}_i$ to be smoothed which corresponds to $\boldsymbol{v}_i$ in the simple smoothing scheme in equation (5.8). Let s_i be the support assigned $\boldsymbol{v}_i$ and $\{\boldsymbol{p}_k\}$ be sampling points inside s_i, and $\{\boldsymbol{n}_k\}$ be those normals, then smoothed and fitted vector $\hat{\boldsymbol{v}}_i$ is obtained as the minimizer of the following minimization problem:

$$\|\mathrm{Lap}\,(\hat{\boldsymbol{v}}_i)\|^2 + \lambda_n \sum_k \sigma_k w_i(\boldsymbol{p}_k)\,\|\hat{\boldsymbol{v}}_i - \boldsymbol{n}_k\|^2 \to \min\,. \tag{5.25}$$

Coefficient λ_n means a priority ratio between smoothing and fitting. It is set to a sufficiently large number, we usually set it at about 10^7. A number σ_k means the confidence level of $\boldsymbol{p}_k$ (details below).

The minimizer of this problem can be analytically obtained in the same manner as the no-fitting version. Let us consider the following simplified form for $x \in \mathbb{R}$:

$$\|ax + b\|^2 + \lambda \sum_k \sigma_k w_i \|c_k x + d_k\|^2 \to \min\,. \tag{5.26}$$

The minimizer can be obtained through an argument similar to the no-fitting version:

$$x = -\frac{ab + \lambda \sum_k \sigma_k w_i c_k d_k}{a^2 + \lambda \sum_k \sigma_k w_i c_k^2}. \tag{5.27}$$

Let us apply this analytical solution to the x component in equation (5.25). The terms a, b, c_k, and d_k in equation (5.27) become as follows:

$$a = -\frac{3}{r_i S_i} \sum_j \phi_{i,j}\psi_{i,j}, \tag{5.28}$$

$$b = \frac{3}{r_i S_i} \sum_j \phi_{i,j} \psi_{i,j} v_{j,x}, \tag{5.29}$$

$$c_k = 1, \tag{5.30}$$

$$d_k = -n_{k,x} \tag{5.31}$$

where $v_{j,x}$ and $n_{k,x}$ are x components of $\boldsymbol{v}_j$ and $\boldsymbol{n}_k$ respectively. Values of c_k and d_k are for each sampling points, while a and b are determined depending on neighboring supports. Note that in the case the fitting effect does not work with value zero for λ, the above minimizer is equal to the one of the non-fitting version. For y and z components, similar equations can be derived. The following is the minimizer represented with vectors:

$$\hat{\boldsymbol{v}}_i = \frac{\left(\frac{3}{r_i S_i}\right)^2 \sum_j \phi_{i,j}\psi_{i,j} \sum_j \phi_{i,j}\psi_{i,j}\boldsymbol{v}_j + \lambda_n \sum_k \sigma_k w_i(\boldsymbol{p}_k)\boldsymbol{n}_k}{\left(\frac{3}{r_i S_i}\right)^2 \left(\sum_j \phi_{i,j}\psi_{i,j}\right)^2 + \lambda_n \sum_k \sigma_k w_i(\boldsymbol{p}_k)}. \tag{5.32}$$

Coefficient vector $\boldsymbol{\alpha}_i$ is updated into the above $\hat{\boldsymbol{v}}_i$.

- Detail Preservation for Constant Terms

  Similar to the updating of coefficient vectors, the update of constant terms can

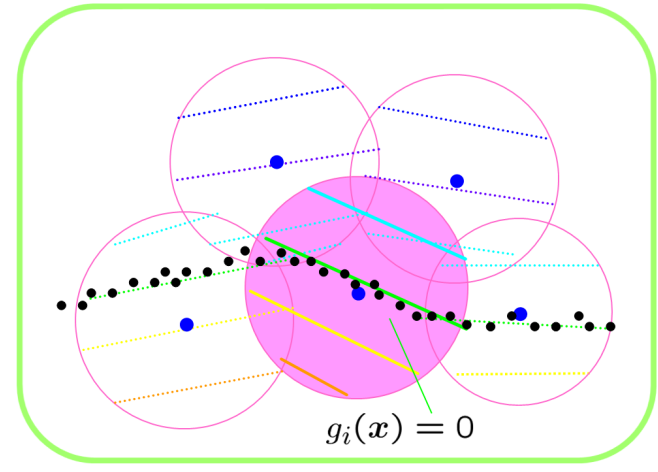


Fig. 5.26. Idea of detail preservation for updating of constant terms.

be described as a minimization problem:

$$\left(\mathrm{Lap}\left(f(\boldsymbol{c}_i)\right) - \mathrm{Div}\left(\hat{\boldsymbol{v}}_i\right)\right)^2 + \lambda_p \sum_k \sigma_k w_i(\boldsymbol{p}_k) g_i^2(\boldsymbol{p}_k) \rightarrow \min, \tag{5.33}$$

where λ_p is a constant introduced for the same purpose as λ_n and set at a sufficiently large value, 10^9 in our implementation. The terms a, b, c_k, and d_k in equation (5.27)

become

$$a = -\frac{3}{r_i S_i} \sum_j \phi_{i,j}\psi_{i,j}, \tag{5.34}$$

$$b = \frac{3}{r_i S_i} \sum_j \phi_{i,j}\psi_{i,j} \left(\beta_j - \boldsymbol{v}_j \cdot (\boldsymbol{c}_j - \boldsymbol{c}_i)\right), \tag{5.35}$$

$$c_k = 1, \tag{5.36}$$

$$d_k = \hat{\boldsymbol{\alpha}}_i \cdot (\boldsymbol{c}_i - \boldsymbol{p}_k). \tag{5.37}$$

Vector $\boldsymbol{\alpha}_i$ is a coefficient vector updated with smoothing and fitting scheme. The minimizer of (5.33), that is the updated constant term is

$$\hat{\beta}_i = \left(\left(\frac{3}{r_i S_i}\right)^2 \sum\nolimits_j \phi_{i,j}\psi_{i,j} \sum\nolimits_j \phi_{i,j}\psi_{i,j} \left(\hat{\boldsymbol{\alpha}}_j \cdot (\boldsymbol{c}_i - \boldsymbol{c}_j) + \beta_j\right) + \lambda_p \hat{\boldsymbol{\alpha}}_i \cdot \left(\sum\nolimits_k \sigma_k w_i(\boldsymbol{p}_k)(\boldsymbol{c}_i - \boldsymbol{p}_k)\right) \right) \Bigg/ \left(\left(\frac{3}{r_i S_i}\right)^2 \left(\sum\nolimits_j \phi_{i,j}\psi_{i,j}\right)^2 + \lambda_p \sum\nolimits_k \sigma_k w_i(\boldsymbol{p}_k) \right). \tag{5.38}$$

- Confidence of Sampling Points

  Confidence level σ_k indicates the reliability of a sampling point $\boldsymbol{p}_k$. In Fig. 5.27, confidence maps of Shiisa-A model are shown. The colors indicate blue for reliable points and red for not-reliable points. As shown in the right image, outliers are assigned low confidence values.

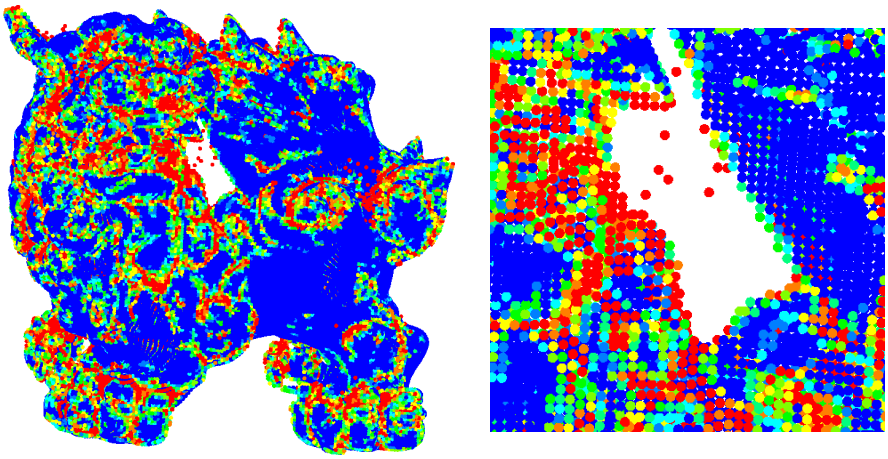


Fig. 5.27. Confidence maps of sampling points for Shiisa-A.

In cases where confidence values are offered with a scanner, using these values is one solution. In other cases where such values are not available, we propose

assigning a confidence value σ_k explained below. This value is defined for each sampling point $\boldsymbol{p}_k$:

$$\sigma_k \equiv \frac{\sum_i w_i(\boldsymbol{p}_k)\tau_i}{\sum_i w_i(\boldsymbol{p}_k)}. \tag{5.39}$$

This means the weighted average of the degree of success of local approximations. Weighting function $w_i(\boldsymbol{x})$ is the same as the one of the spherical covering. The degree of success of local approximations τ_i is measured for each support in reference to the distribution of sampling points inside a support. Here is the definition of τ_i for a support s_i:

$$\tau_i \equiv \exp\left(-2\omega_i^2\right) \tag{5.40}$$

where ω_i is the angle between the eigenvector $\boldsymbol{N}_i$ with the smallest eigenvalue obtained with PCA for $\{\boldsymbol{p}_k\}$ and, the vector from the centroid $\boldsymbol{G}_i$ of $\{\boldsymbol{p}_k\}$ to the support center $\boldsymbol{c}_i$.

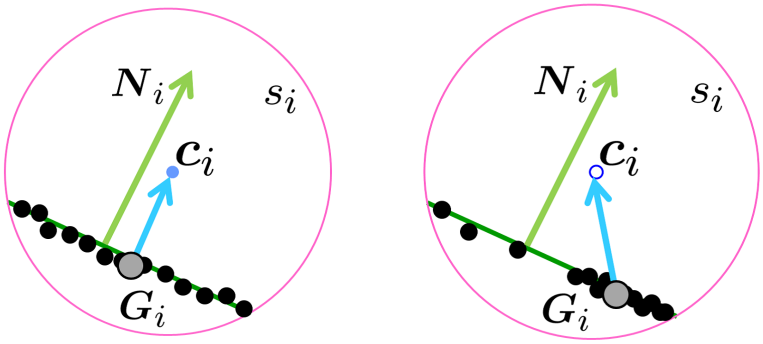
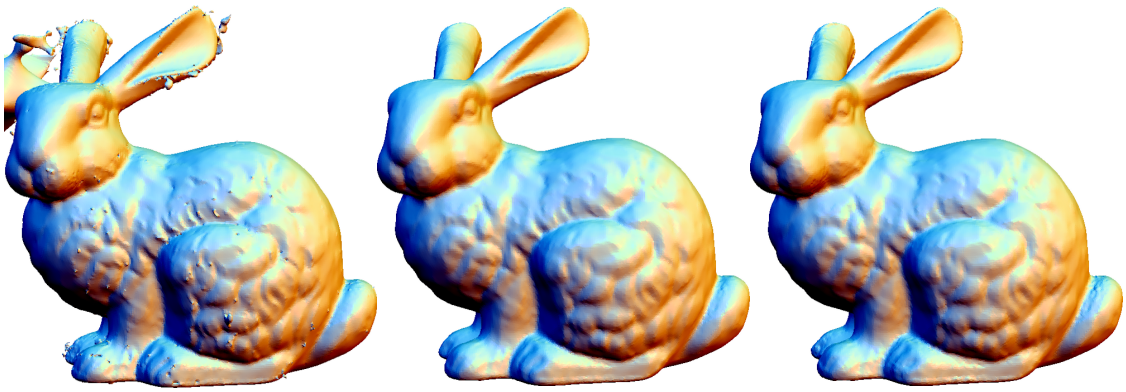


Fig. 5.28. Idea for confidence value σ_k.
Left: spherical support with uniformly sampled points.
Right: spherical support with non-uniformly sampled points.

See Fig. 5.28, examples of 2D spherical supports with sampling points (black dots). The left image shows a spherical support with uniformly sampled points and the right image shows non-uniformly sampled case. In each images, the approximation line for $\{\boldsymbol{p}_k\}$ obtained with PCA is shown in dark green, and corresponding eigenvector $\boldsymbol{N}_i$ is shown with a light green arrow. This vector $\boldsymbol{N}_i$ is an approximated normal of $\{\boldsymbol{p}_k\}$ in terms of sphere s_i [HDD+92]. The centroid $\boldsymbol{G}_i$ of $\{\boldsymbol{p}_k\}$ is indicated with a large gray point. When the sampling points distribution is uniform, vectors $\boldsymbol{N}_i$ and $\boldsymbol{c}-\boldsymbol{G}_i$ have almost the same direction, and the value of τ_i increases. If the sampling density changes inside s_i, their directions are different each other and τ_i decreases. For a sampling point near a lack of sampling, the value of σ_k becomes small.

- Convergence

  Introducing the fitting terms helps smoothing to converge quickly. As can be observed for Fig. 5.29, the results from five iterations of smoothing, (b), have no notable difference with results from twenty iterations, (c).



(a) (b) (c)

Fig. 5.29. Convergence of smoothing iteration.
(a) initial model obtained with partition of unity approach.
(b) results with five times iteration of smoothing.
(c) results with twenty times iteration of smoothing.

Algorithm Summary

Here we summarize the surface reconstruction algorithm using diffusion technique.

Algorithm: surface reconstruction using diffusion technique

1. Generate a spherical cover for the bounding box through partition of unity
2. Evaluate the confidence of the sampling points
3. Iteratively apply smoothing of partition of unity implicits preserving details
4. Polygonize the zero-level surface

Step 1 is performed as explained in Section 3.4.1. Steps 2 and 3 are newly described in this section. The number of the iterations in Step 3 is five in our experiments for all models in this thesis. In the last step, the zero-level surface is polygonized using the SurfaceNets [Gib98] with uniform sampling around only the zero-level surface as proposed in [Blo88]. We set a sampling point with a high confidence value as a seed point for detecting the zero-level surface.

Experimental Results

Parameter setting. All input data in this section 5.2.3 (except Fig. 5.4, 5.31, and 5.36) are noisy raw scanned data. All parameters except ε are fixed unless the settings are described. That is, the only free parameter required by the diffusion algorithm is approximation error tolerance ε, which depends on the noise level specified by the scanning device. A comparison of results with different values of ε is shown in Fig. 5.30. For the details of ε, see Section 3.4.1.

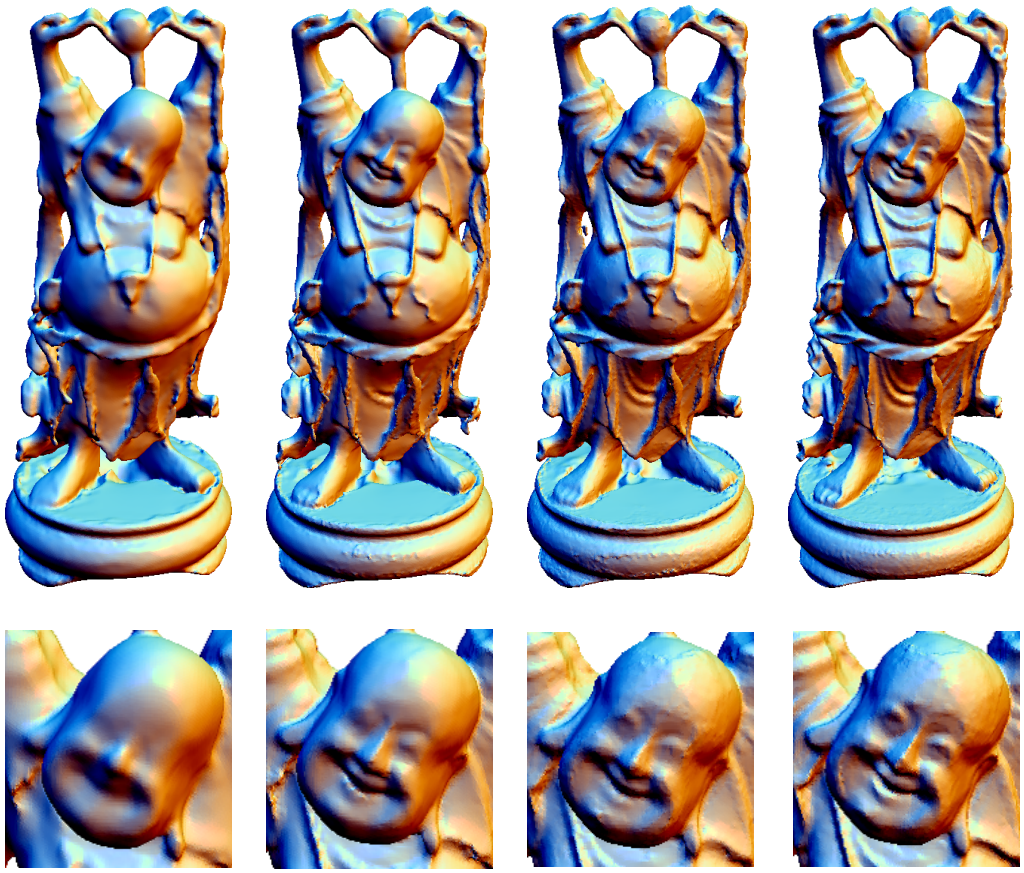


Fig. 5.30. The reconstructed meshes with different values of ε.
From left to right, $\varepsilon = 1.0 \times 10^{-2}, 5.0 \times 10^{-3}, 2.5 \times 10^{-3}$, and 1.25×10^{-3}.

Noise robustness. The robustness for noise is demonstrated in Fig. 5.31. The sampling points are obtained from meshes added Gaussian noise. The noise level on the right model is twice as large as that on the left model. The value of the standard deviation of the noise on the left/right model is a quarter/half of the averaged edge-lengths. The reconstructed meshes are topologically correct even for data with large amount of noise.

In Fig. 5.32, the robustness for normal noise is demonstrated. For data with normals randomly rotated by 30°, our approach succeeded in reconstruction (b). For a rotation

angle of 60°, the reconstructed model is affected in terms of smoothness (d). With setting λ_n to zero, even for such cases, a smooth surface is obtained (e).

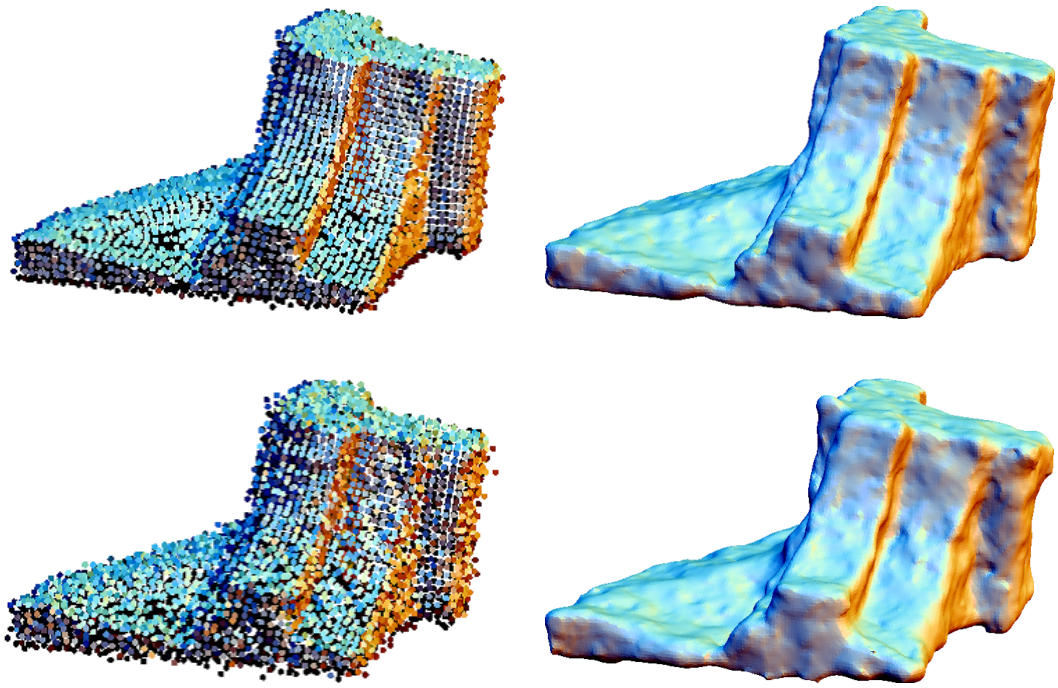


Fig. 5.31. Noise robustness.
Left: sampling points obtained from noisy meshes.
Right: reconstructed surface meshes.

Performance. Timing results are reported in the Table 5.1. In order to perform PU smoothing efficiently, we construct a graph of adjacency of supports and store it in a list data structure.

Following our observation, the average of degrees of the sphere intersection graph is about 100, which makes processing large data sets difficult. According to our experiments, reducing the degree with shrinking the radii of supports to 0.7 times the original radii reduces the computational time and space. The average of degree after this shrinking is about 40. The results obtained with this shrinking are not different from those without shrinking. This shrinking is only for constructing a sphere intersection graph, so we use the original radii in all the other processes including the calculation of $\phi_{i,j}$.

Comparison. We compared our results with MPU [OBA+03] and Poisson surface reconstruction [KBH06] for a Shiisa-A model, see Fig. 5.33. In the MPU results, the peak on its beard stretches and extra components appear around the tail. Poisson surface reconstruction gives a surface on which details are lost because of its smoothing effect. On the

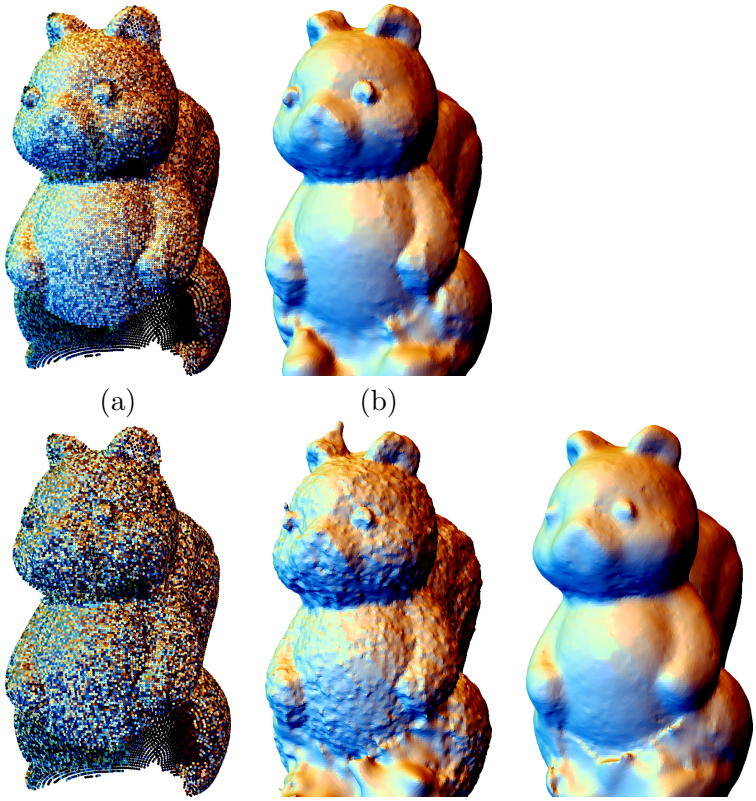


Fig. 5.32. Normal noise robustness.
(a) Data with normals rotated randomly by 30°.
(b) Reconstructed mesh for sampling points in (a).
(c) Data with normals rotated randomly by 60°.
(d) Results for sampling points in (c).
(e) Results for sampling points in (c) with $\lambda_n = 0$.

Model	# pts.	ε ($\times 10^{-3}$)	# supps.	Time [min:sec] init.	smooth.	poly.	Peak RAM	# tris.
Bunny	362K	2.5	176K	0:18	0:19	0:14	74M	315K
Buddha	3.2M	2.5	540K	2:05	2:26	0:58	234M	1.07M
Buddha	3.2M	1.25	2.94M	3:41	13:35	1:08	1.23G	1.08M
Armadillo	2.3M	1.0	7.35M	5:42	36:21	3:49	3.24G	3.30M

Table. 5.1. Computational time and performance of surface reconstruction.
From left to right: the number of sampling points, value of ε, the number of supports, computation time for initialization of PU, for smoothing, and for polygonization, the maximum memory space which our algorithm used, and the number of triangles of the result mesh.

other hand, the proposed method generates a smooth surface with the details preserved well, see the close-ups in the lower columns of Fig. 5.33.

Another comparison for a large data set with Poisson surface reconstruction is shown in Fig. 5.34. The reason that Poisson surface reconstruction is selected as a comparative

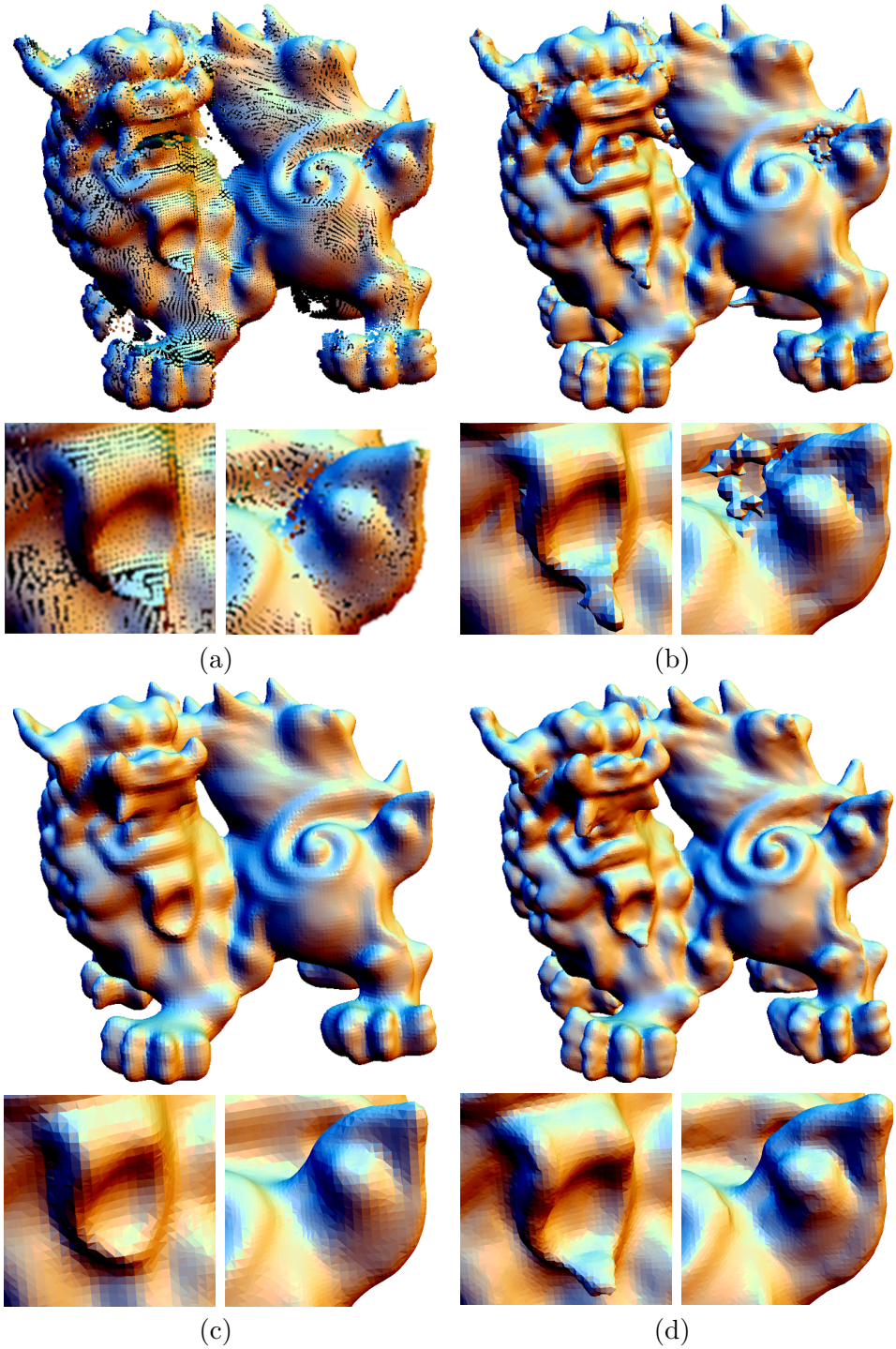


Fig. 5.33. Comparison of surface reconstruction method using diffusion technique with other methods.
(a) Sampling points. (b) Results with MPU.
(c) Results with Poisson surface reconstruction.
(d) Results with surface reconstruction method using diffusion technique.

algorithm is that it has excellent reconstruction capabilities, and the software required is freely available. The parameters are, for Poisson surface reconstruction, set to 11 for the

level of the octree, and for ours, $\varepsilon = 1.0 \times 10^{-3}$. Although both methods work well, in the close-ups of hands, it is observed that details are better preserved with our method (see the bumps in hands). We consider this is because the proposed algorithm uses first- and zeroth-order approximation, while Poisson surface reconstruction uses only zeroth-order approximation.

Another example is shown in Fig. 5.35. The surface is gradually sparsely sampled. The results with Poisson surface reconstruction shrink near the tail, but the results generated with proposed method represents a rough shape of the tail.

One distinct property of the diffusion technique is that reconstructed surfaces tend to extend in the tangential directions. See the tail of the dragon in Fig. 5.35. A more clear comparison of this property can be seen in Fig. 5.36. The sampling points are vertices of parts of a noisy octahedron. The results obtained with Poisson surface reconstruction tend to be rounded near areas where there is a lack of sampling, because of shrinking problem. On the other hand, in the results from the proposed method, each face extends in the tangential directions. This extension is caused by diffusion of normals. In contrast, Poisson surface reconstruction solves Laplacian equation on the region where sampling is missing. Thus the missing parts are filled with minimal surfaces.

5.2.4 Integration of Graph-cut and Diffusion

Introduction

In this section the integration of Graph-cut and diffusion algorithms is proposed. As mentioned above, Graph-cut provides outlier-robustness and diffusing local approximation functions gives noise-robustness, so combination of these two algorithms is expected to handle low quality data well. We use the results of Graph-cut to determine inconsistent spherical supports which are considered wrongly approximating. And then the smoothing algorithm diffuses the local approximation functions of consistent spherical supports allover the covered area.

Algorithm

The overview of this integrated algorithm is below. The first and the last steps are completely the same as the previously explained ones. The Graph-cut and the diffusion are basically same, and the difference parts are described in the below paragraphs.

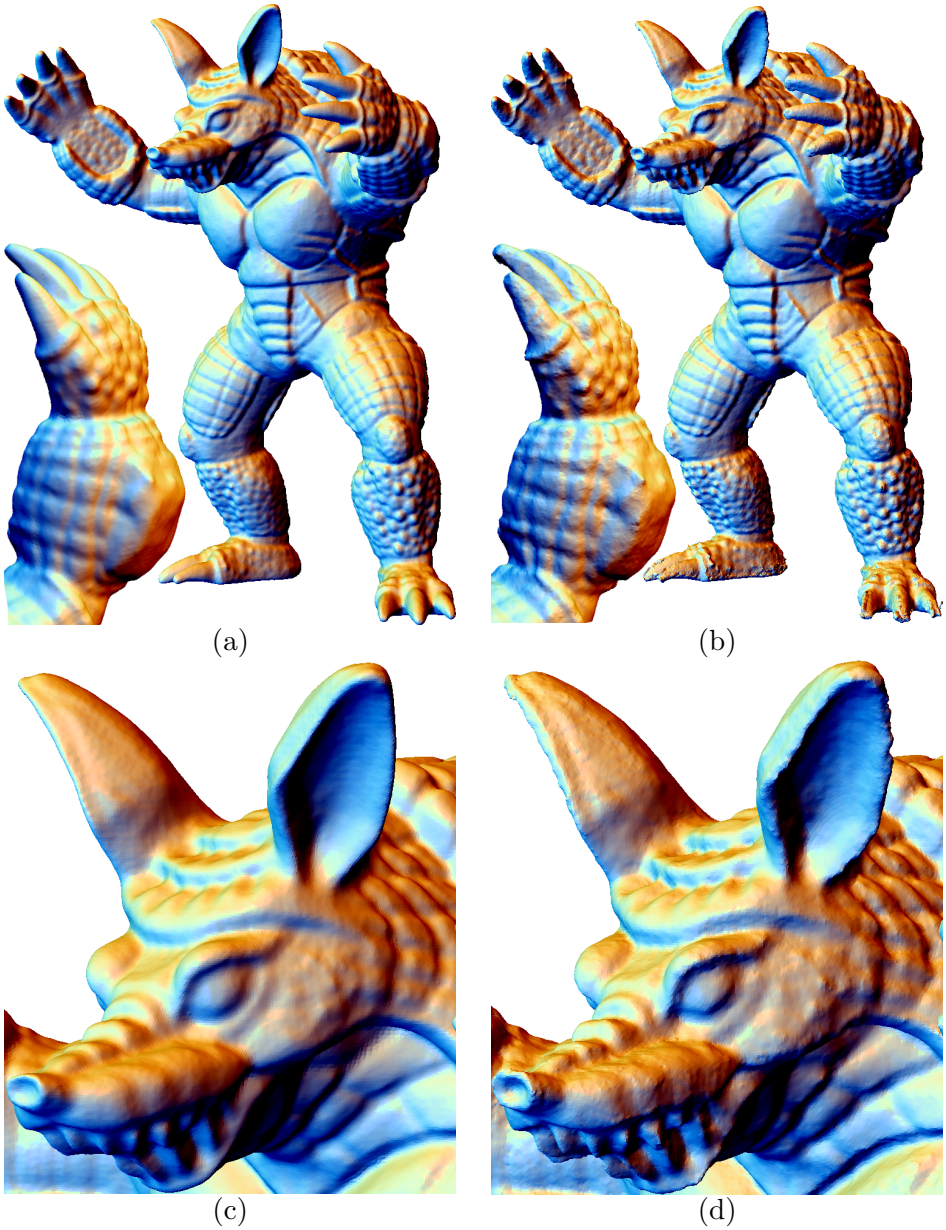
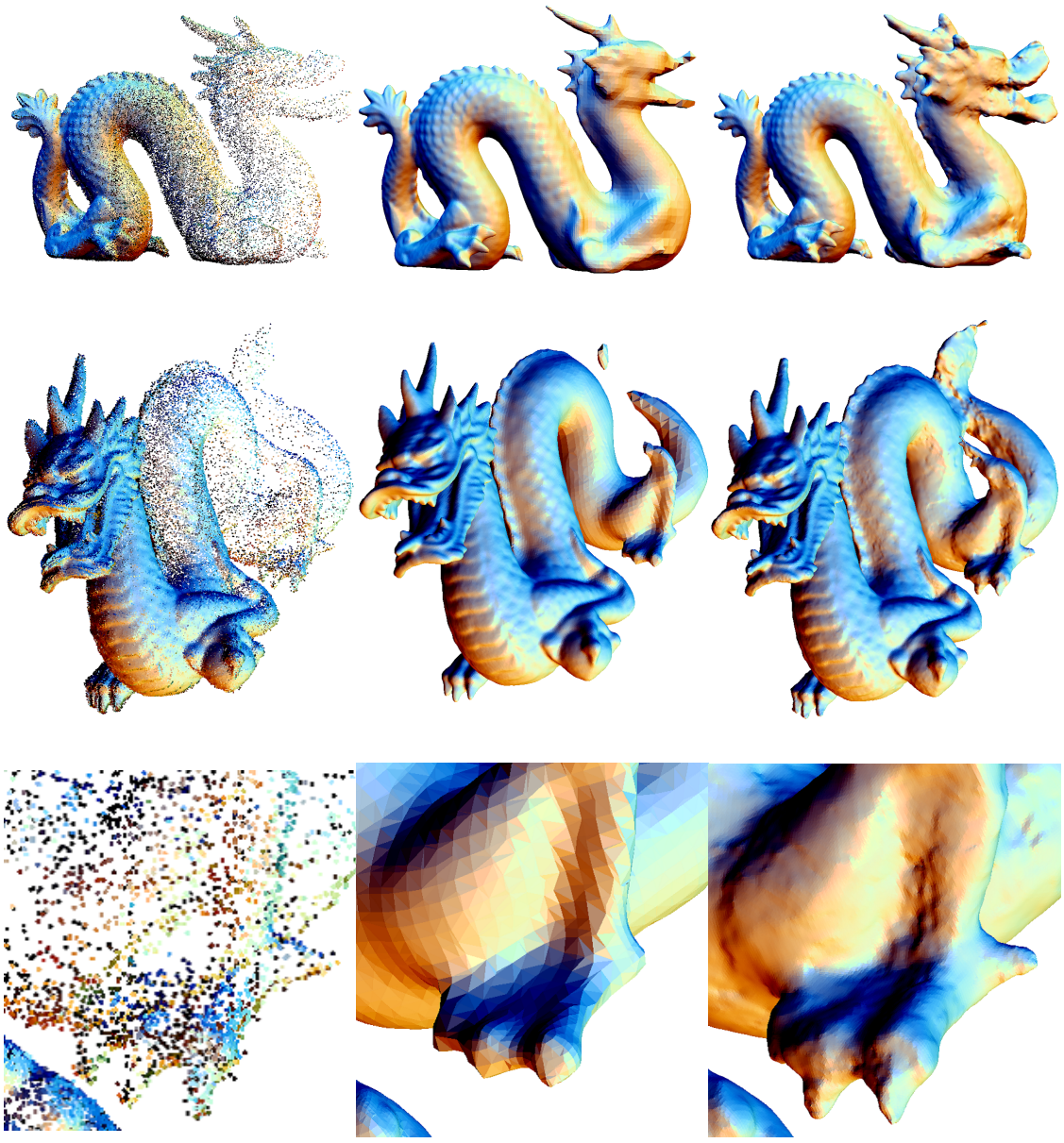


Fig. 5.34. Comparison with Poisson surface reconstruction.
Left: Results from the Poisson surface reconstruction.
Right: Results from smoothing of local approximation functions.

Algorithm: surface reconstruction with Graph-cut and diffusion

1. Generate a spherical cover through Partition of Unity
2. Operate the Graph-cut and change some local approximation functions into zero functions
3. Diffuse the local approximation functions of the nonzero local approximation functions
4. Polygonize the zero-level set



(a) (b) (c)

Fig. 5.35. Comparison of results for gradually down-sampled points. Lower column displays close-ups of the models in the middle column.
(a) Sampling points. (b) Results with Poisson surface reconstruction. (c) Results with smoothing of local approximation functions.

Graph-cut for integration The only difference with the algorithm in Section 5.2.2 is the graph. In the previous Graph-cut algorithm, graph is constructed using the tetrahedral mesh obtained from the sphere intersection graph (SIG). In this integrated algorithm, the SIG serves as the base structure of the graph as the tetrahedral mesh in the previous algorithm. It is needed for the all supports to be classified in the following smoothing step,

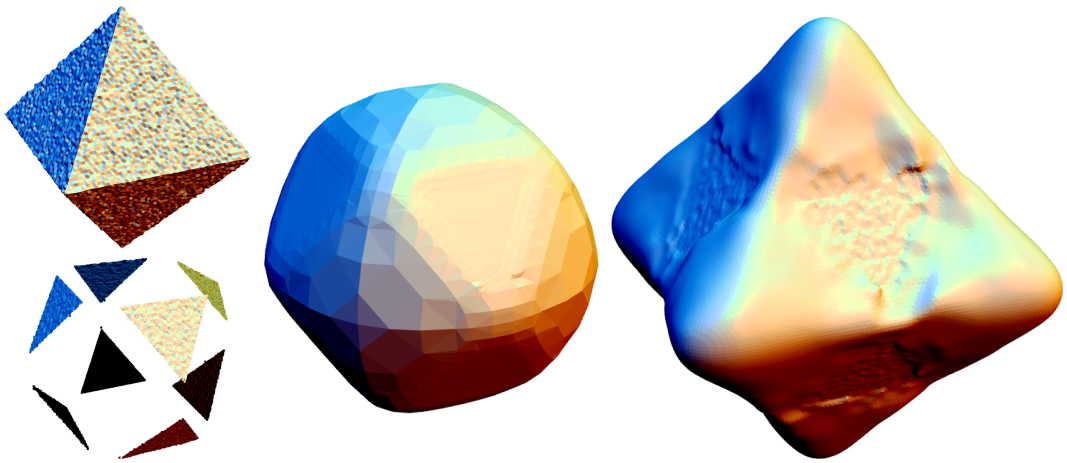


Fig. 5.36. Comparison of the properties of reconstructed surfaces.
Left: the original noisy mesh (top) and partially sampled points with normals from the above mesh (bottom). Middle: results generated by Poisson surface reconstruction. Right: results generated by smoothing approach for local approximation functions.

so the tetrahedrization is not suitable for this graph generation (tetrahedrizaion may omit several supports). This change is reasonable since a polygonization grid structure is no longer needed to be generated in this step. The polygonization is performed with another scheme in the latter step, maybe using another grid structure. In addition, since the tetrahedral mesh generation is pretty time-consuming, so it also saves lots of calculation time.

With the results of Graph-cut, the spherical supports which the signs of approximation function at the centers are different from the results of inside/outside classification through the Graph-cut can be recognized. In this thesis, such supports are referred as inconsistent supports.

Diffusion for integration After the Graph-cut, the local approximation functions associated with inconsistent supports are changed into the zero functions. As a result, only the local approximation functions of the consistent supports can be diffused. The diffusion scheme basically follows the algorithm in Section 5.2.3. The values of confidence of inconsistent supports are set to zero and, on the other hand, the other supports has the same confidence values as the ones proposed in the equation (5.40). So the new definition of the confidence of a support s_i is

$$\tau_i = \begin{cases} \exp\left(-2\omega_i^2\right) & (s_i \text{ is consistent}) \\ 0 & (s_i \text{ is inconsistent}) \end{cases} . \tag{5.41}$$

Experimental Results

Here are some experimental results obtained from the above integration. The results show that the classification through Graph-cut can handle low quality data well.

The first example is for scanned data with 1161580 points including 50 outliers. The points except the outliers are obtained with scanning the figure in the left image of Fig. 5.37. The points added 50 outliers are shown in the right image. The scanned points include web-like clouds of outliers between the forefoot the ball. Fig. 5.38 shows the reconstructed surfaces for this data. In the result with only the diffusion technique, large extra surfaces appear around the cat. With the help of the Graph-cut, such surfaces do not appear any more. And note that the forefoot and the ball are correctly reconstructed.

The images in Fig. 5.39 show the results of Graph-cut for the centers of supports. The left image shows the centers of the all supports. This means the unclassified state. Applying the Graph-cut, the centers are classified into inside or outside of the object. The inside-classified centers are shown in the middle image. It can be seen that the clouds of outliers are removed. The right image shows the inconsistent centers. The results of recognition of outliers with thresholding the confidence values (for all data in this chapter, we generally set 0.007) of the input points are shown in Fig. 5.40. The outliers around the cat figure are recognized well.

Fig. 5.41 shows the results for gradually downsampled data with 294259 points including 100 outliers. Even the head, sparsely sampled area, is well reconstructed.

The images in Fig. 5.42 are the results for sampling points with large lacks of sampling and 20 outliers. The number of the total points is 28751. The result with only diffusion technique, extra surfaces are generated near the back. With the aid of the Graph-cut, the smooth shape is reconstructed. With comparing with the result with only the Graph-cut and bi-Laplacian smoothing, it can be observed that this integration gives more smooth shapes.

5.3 Discussion

Spherical Cover

Normal Dependency. A drawback of this process is a necessity of normals. In the case many normals are flipped proposed algorithm cannot offer a correct result, although it can handle data with a certain amount of noise.

Graph-cut

Time Consumption in Tetrahedrization. One bottleneck of Graph-cut approach is time-consumption. For processing a few million points, our surface reconstruction using com-

Fig. 5.37. A cat figure and its scanned points added outliers.

bination of partition of unity and Graph-cut takes a few tens of minutes, while both Poisson surface reconstruction and MPU work on several minutes. The most expensive procedure is tetrahedrization. We plan to improve this operation in our future work.

Determination of parameter k. Another issue is finding a proper value of k in equation (5.1). Fig. 5.43 shows this problem. With value $k = 2.3$, the left arm does not appear while the other parts are well reconstructed. In our current algorithm, the value of k is fixed however, results in Fig. 5.43 suggest to make the value of k adaptive to the complexity of the object or the density of sampling points.

Diffusion of Local Approximation Functions

Validity of Linearity of Local Approximations. For smoothing local approximations with diffusion technique with the aid of discrete differential operators, we made assumption that local approximations are linear. This approximation may be considered as a limitation, but linear approximation is sufficient for surface reconstruction because its output generally takes the form of a polygonal mesh. Using local linear approximations tends to generate more supports than the quadratic approximations of MPU [OBA+03], but in terms of noise robustness, algorithms employing linear approximation are better than those with high-order approximations.

Expansion of Local Approximations. The linearity of local approximations is not a limitation for surface reconstruction. But it is sure that employing various functions as local

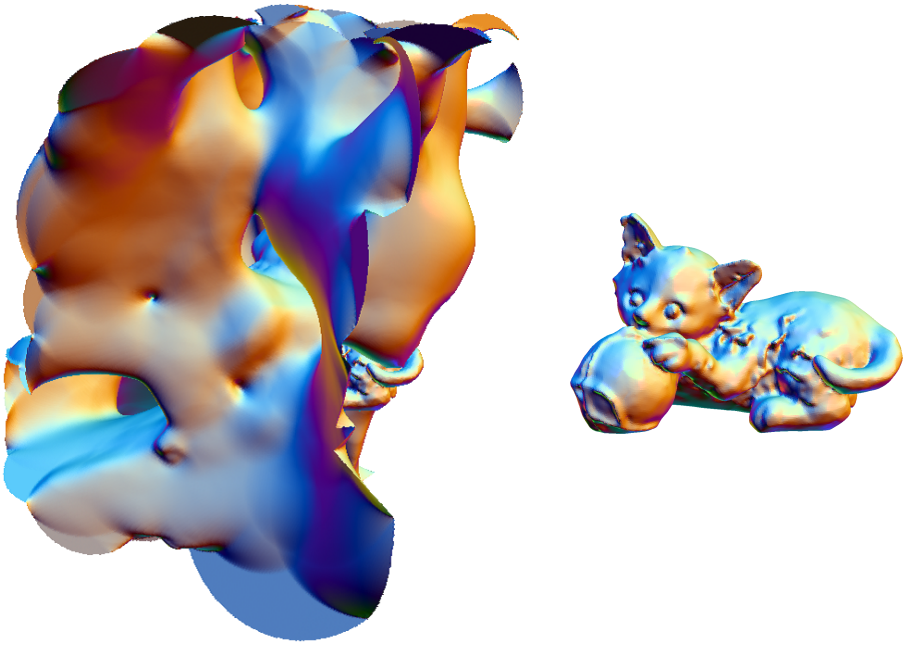


Fig. 5.38. Surfaces obtained with the integration of the Graph-cut and diffusion technique.
Left: surfaces obtained with only the diffusion.
Right: surfaces obtained with the Graph-cut and the diffusion.

approximations helps proposed smoothing to be used in more applications. So it is worth changing the primitive functions to be quadratic and cubic functions. For this purpose, redefining the discrete operators is needed. For a related issue, see the discussion in Section 4.5.

Exclusivity of Support Centers. In updating local approximations, we assume the centers of supports are not included in other supports. Good results are obtained under this assumption however, we observed that most centers actually exist in more than one supports. The influences of this assumption requires more research. We consider that the spherical cover generation based on an octree helps generation of good results.

Representation of Sharp Feature. We proposed anisotropic smoothing which roughly preserves edges, but more precise representation is needed in many applications. Representation of sharp features such as edges and corners is another issue.

Time Consumption in Diffusion. One drawback of the iterative diffusion technique is that the proposed method is three–five times more time-consuming than that of the Poisson surface reconstruction. However, our PU smoothing based on iterative averaging can be easily parallelized by using multi-core CPU.

Fig. 5.39. Results of the classification with Graph-cut for the centers of supports.
Left: the centers of the all spherical supports.
Middle: the centers of supports classified into inside.
Right: the centers of inconsistent supports.

Fig. 5.40. Results of the classification using the confidence values.
Left: the input points with high confidence values.
Right: the input points with low confidence values.

Processing Higher-Order Data. Possible applications of proposed diffusion technique include volume processing and the smoothing of more attributed vectors such as 4D vectors and texture data. Proposed diffusion technique can handle these higher-order data as well as 2D and 3D data demonstrated in this thesis.

Integration of Graph-cut and Diffusion

Parameter Setting. For the integration of the Graph-cut and diffusion, the parameters are set as follows:

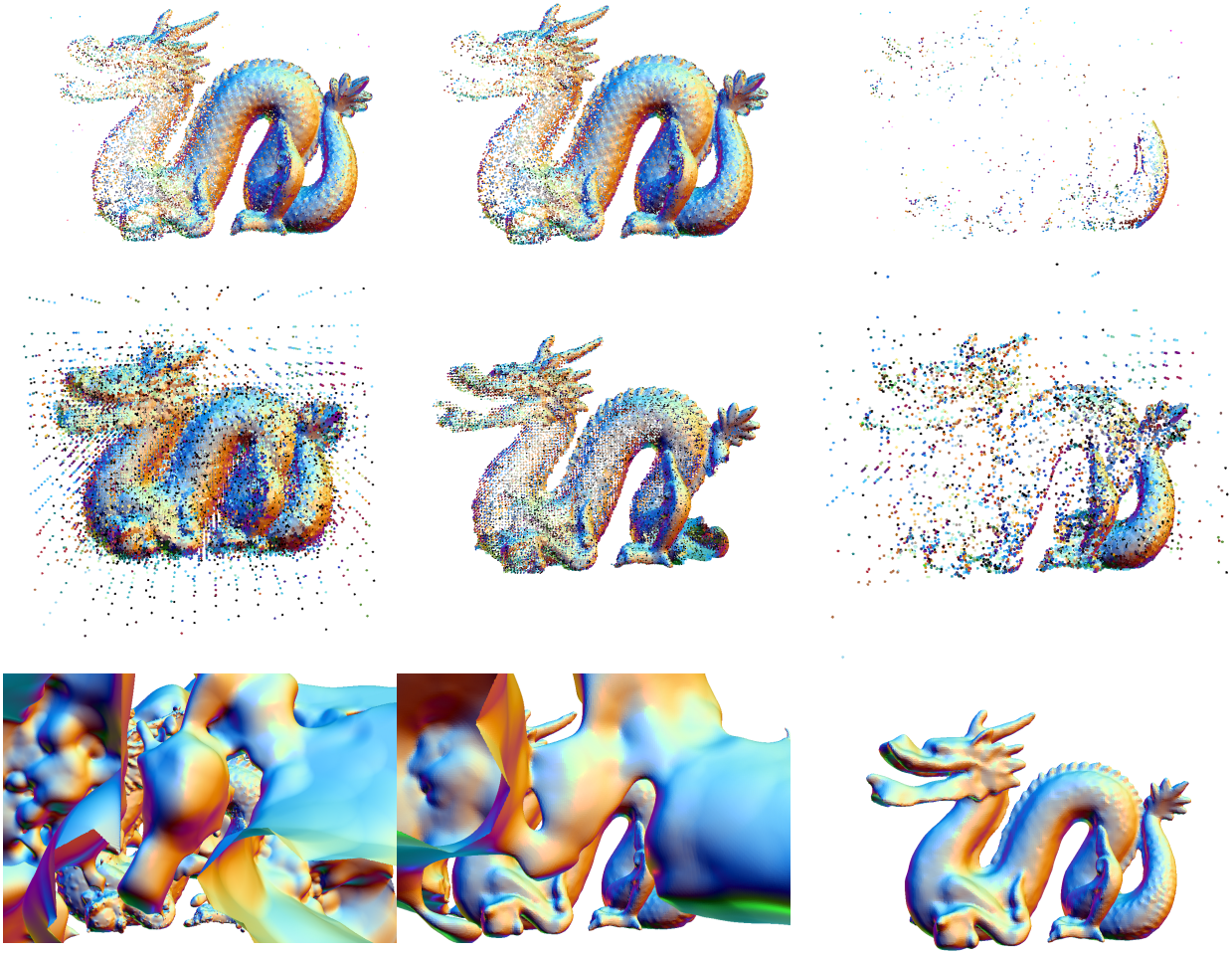


Fig. 5.41. Results of the integration approach for gradually sampled and outlier-added points.
Upper: input points. From left to right, the all points, the points with high confidence values, and the points with low confidence values.
Middle: the centers of the spherical supports. From left to right, for the all supports, for the supports classified into inside, and for the inconsistent supports.
Lower: reconstructed surfaces. From left to right, surfaces obtained with only PU, surfaces obtained with diffusion, and surfaces obtained with the Graph-cut and the diffusion.

- approximation error tolerance ε: 2.0×10^{-3} for the data in Fig. 5.37 and Fig. 5.41, and 5.0×10^{-3} for the data in Fig. 5.42.
- k of the Graph-cut: a value between 27 and 30.
- fitting parameters λ_n and λ_p for diffusion: 1.0×10^7 and 1.0×10^9 respectively.

These settings cause some lost of details as shown in the final surfaces in Fig. 5.41. The scales in the neck and the details of the face are smoothed. This is because that the settings of the approximation error tolerance and fitting parameters. If these values are

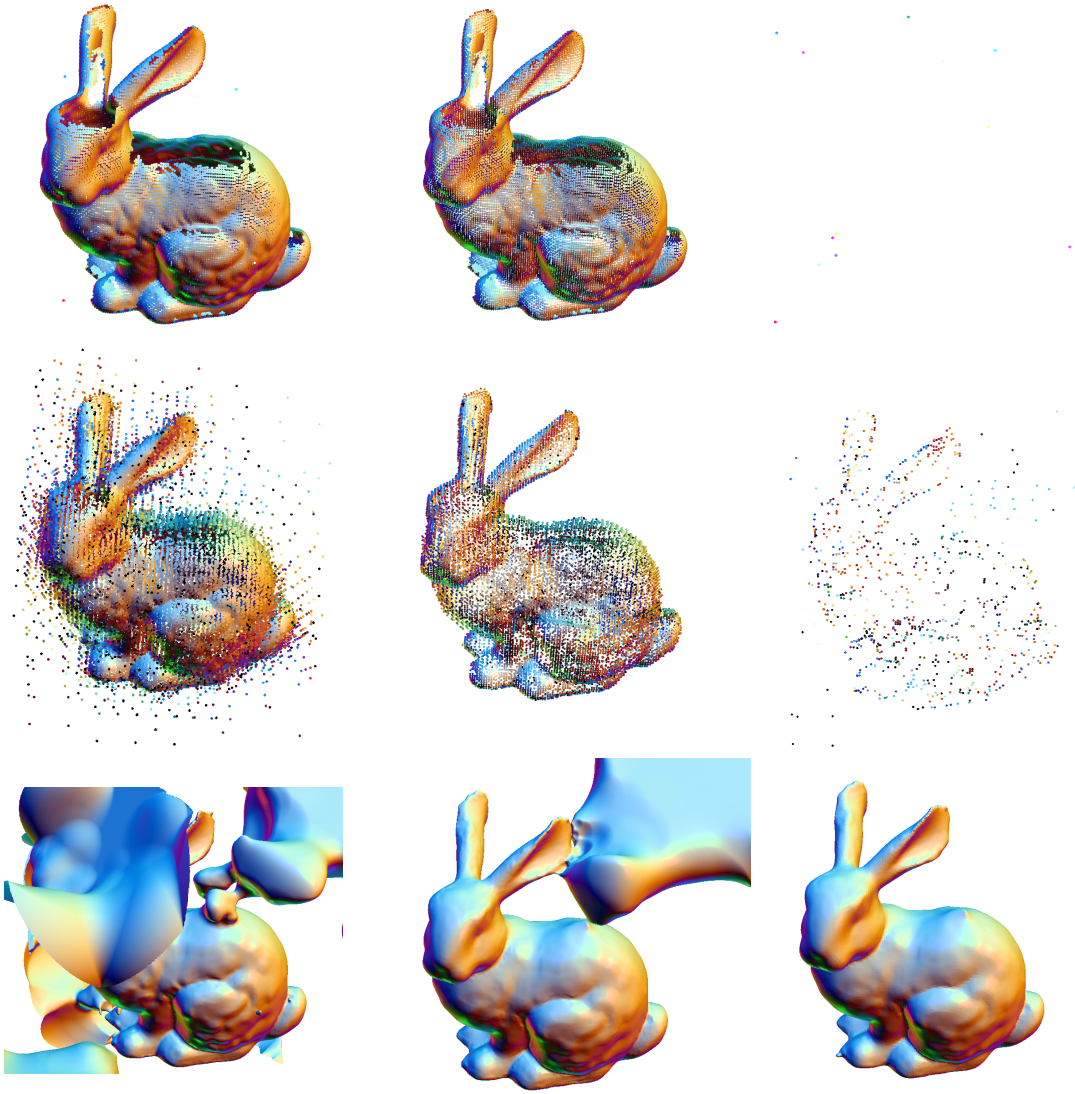


Fig. 5.42. Results of the integration approach for sampling points with lacks and outliers.
Upper: input points. From left to right, the all points, the points with high confidence values, and the points with low confidence values.
Middle: the centers of the spherical supports. From left to right, for the all supports, for the supports classified into inside, and for the inconsistent supports.
Lower: reconstructed surfaces. From left to right, surfaces obtained with only PU, surfaces obtained with diffusion, and surfaces obtained with the Graph-cut and the diffusion.

set to more strict values, the details will be reconstructed, but extra surfaces may exist. The images in Fig. 5.44 shows results with $\varepsilon = 2.0 \times 10^{-3}$, and $\lambda_n = 1.0 \times 10^7$ and $\lambda_p = 1.0 \times 10^9$ respectively. The details are better represented than the results in Fig. 5.42, but undesired surfaces remain. This is a trade-off between precise representation and no-extra-component reconstruction. In the case that precise representation is required, post-processing to remove undesired surfaces will be needed.

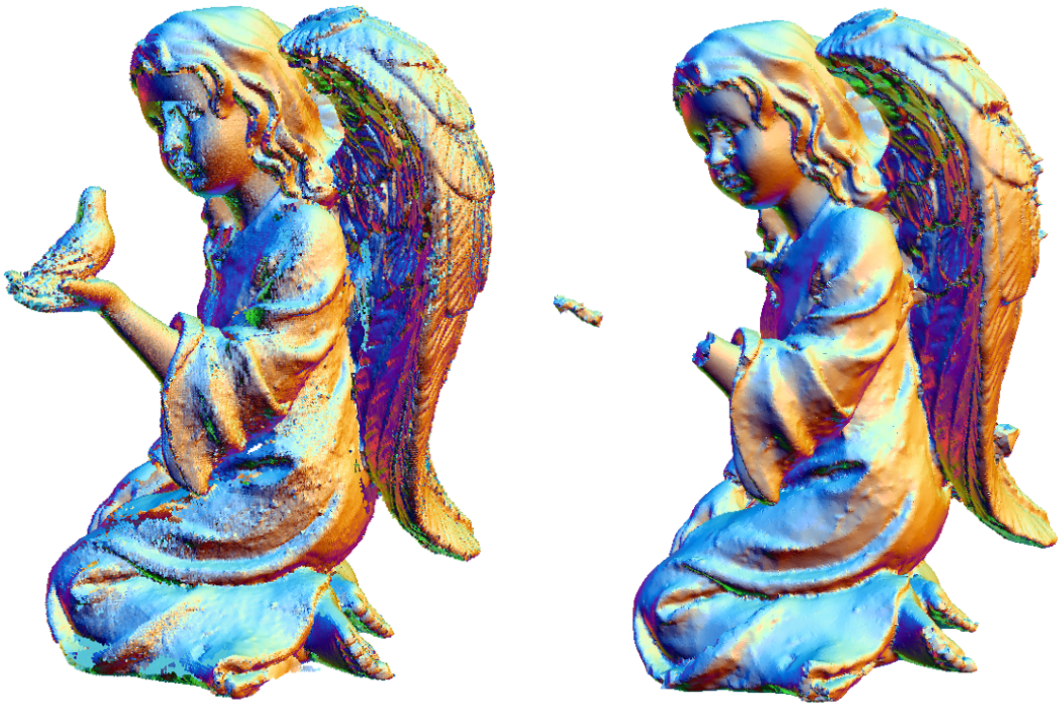


Fig. 5.43. Problems caused by fixing value of k in equation (5.1).
Left: sampling points. Right: reconstructed model with $k = 2.3$.

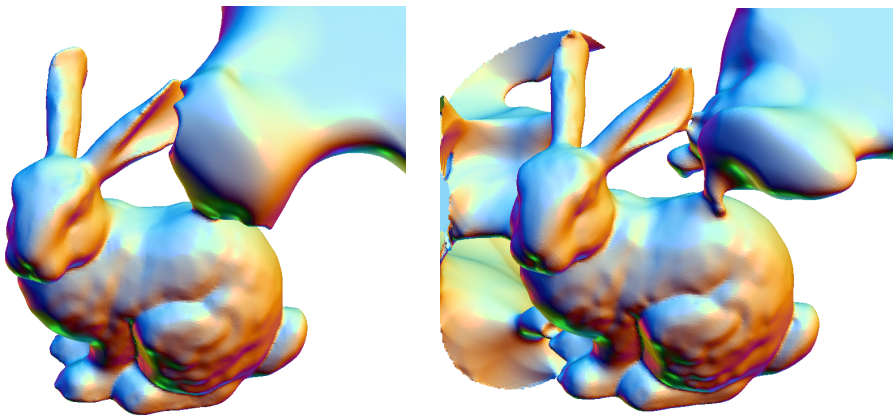


Fig. 5.44. Results of the integrated approach with different parameters.
Left: $\varepsilon = 2.0 \times 10^{-3}$.
Right: $\lambda_n = 1.0 \times 10^{7}$ and $\lambda_p = 1.0 \times 10^{9}$.

5.4 Summary

In this chapter, we proposed a surface reconstruction algorithm as an application of a spherical cover with differential operators. Proposed algorithm has advantages of noise robustness and detail-preservation. Robustness for outliers is achieved with combination of partition of unity approach and Graph-cut. Robustness for noise is mainly achieved with

the aid of a diffusion of local approximation functions which use the discrete differential operators. Robustness for lack of sampling is realized with both proposals. In addition, a measure of confidence of sampling points specialized for a spherical cover is also proposed.

The local approximations in this algorithm are linear functions. Changing these functions to various functions is one of future tasks to be performed, and we also plan to preserve sharp features.

Chapter 6

Skeletal Structure Generation

Skeletal structures are 2D skeletal sheets or 1D skeletal curves obtained with simplifying a 3D solid. Analysis with skeletal structures helps us recognize object's shape intuitively and performed especially in areas involving deformation, recognition, and matching. In this thesis we treat only 2D skeletal sheets as target skeletal structure and simply refer to such a structure as skeletal structure or skeletal sheet. In this chapter, we propose an algorithm to generate a surface mesh robustly and directly from noisy CT-scanned data. This algorithm consists of approximation technique and analytical differential calculations, both of which are based on a spherical covering technique. Because of employing these two techniques, proposed algorithm achieves noise-robust skeletal structure extraction.

6.1 Introduction

Skeletal-sheet plays important roles in many areas including industrial engineering, computer graphics, medical engineering. Among these areas, we consider the case of use in industrial engineering such as reverse engineering. This area treats mainly thin objects and in many cases these shapes are obtained as CT-scanned data. Recent advances in X-ray-based CT scanning technology have enabled accurate measurement even for mechanical objects consisting of thin parts. A thin object and its CT-scanned data are shown in the images of Fig. 6.1 (a) and (b). We can easily extract the boundary of a scanned object using isosurface extraction techniques such as Marching Cubes and dual contouring [LC87, JLSW02]. An example of extracted isosurface of the object can be seen in Fig. 6.1 (c).

However, as mentioned in [FKS06, FSKK05], it is often necessary to extract a scanned object as single sheet as a structure in Fig. 6.1 (d) (rather than as a two-sided thin object in Fig. 6.1 (c)) for further digital manipulation such as reverse engineering, quality evaluation and physical simulation. Skeletal-sheet extraction is one possible solution to meet this request. Extracted sheet is mainly represented as a surface mesh.

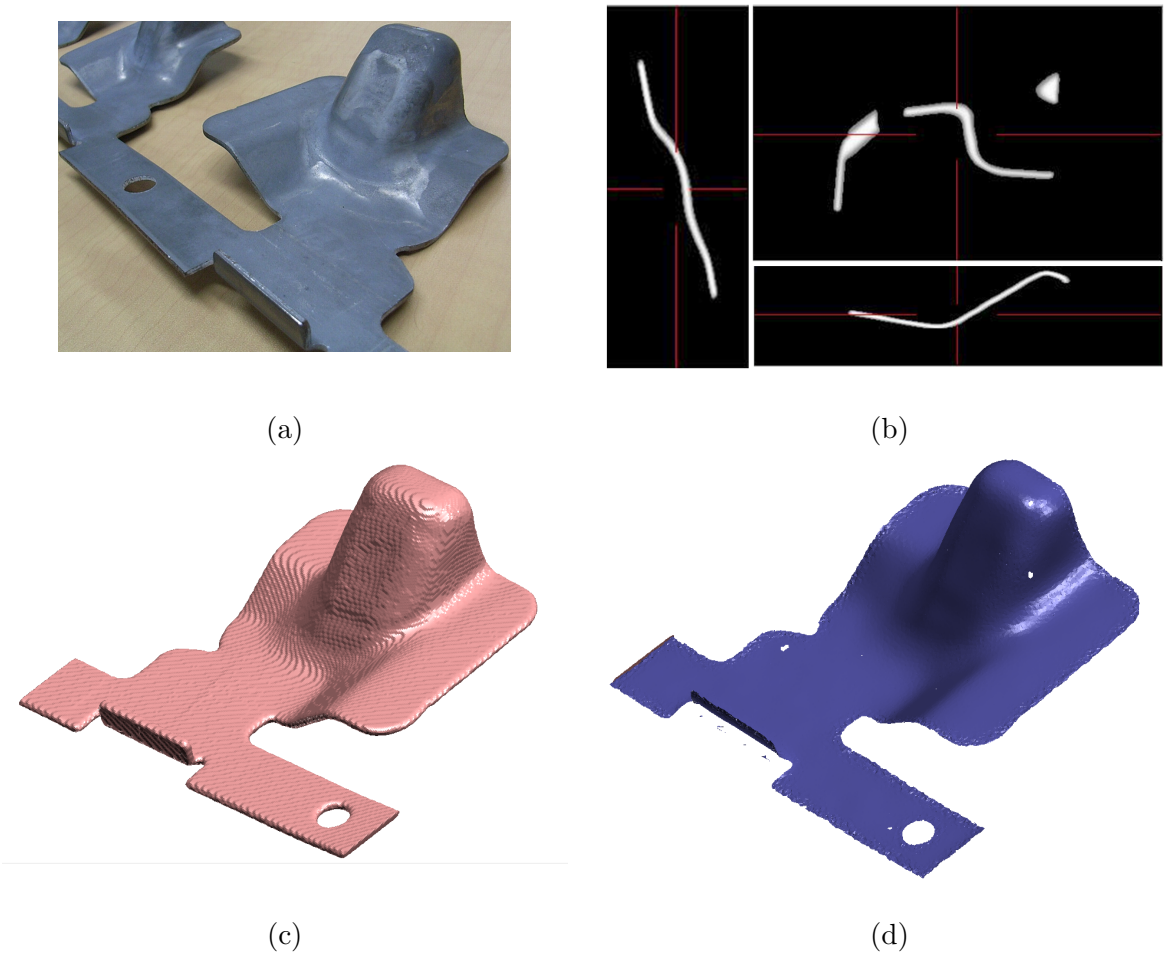


Fig. 6.1. Industrial object: CT-scanned data, extracted isosurface, and skeletal sheet.
(a) Not divided industrial objects.
(b) Three slices of CT-scanned data of an object which is a unit of the object in (a).
(c) Isosurface of a CT-scanned data of the object.
(d) Extracted skeletal sheet.

6.1.1 Problem Setting

The goal of this study is to extract a 2D skeletal structure as a surface mesh approximating it directly from CT images of an object. As a target object, we treat only a thin object which is made up of single material like that demonstrated in Fig. 6.1. Generally, a volumetric image of a real-world object obtained using a scanning technology such as the CT scan method includes much amount of noise. The difficulties of this problem exist in achieving robustness for noise in CT-scanned data.

6.1.2 Our Approach

Given a volumetrically sampled solid object, our method extracts a well-connected and not-fragmented skeletal structure represented as a polygonal mesh.

The CT canned value of an object increases gradually near the boundary of the object. In the direction of thickness, CT scanned value of a thin object increases then decreases without reaching sufficiently high value, as a result, a peak appears near the center. See Fig. 6.2 which shows three objects with different thicknesses. CT value is affected by the thickness of objects. The object on the top is sufficiently thick and CT value reaches high value. As object becomes thinner, the range which achieves high value decreases. For object which is not sufficiently thick as seen in the bottom image, its CT value decreases soon without reaching an expected value.

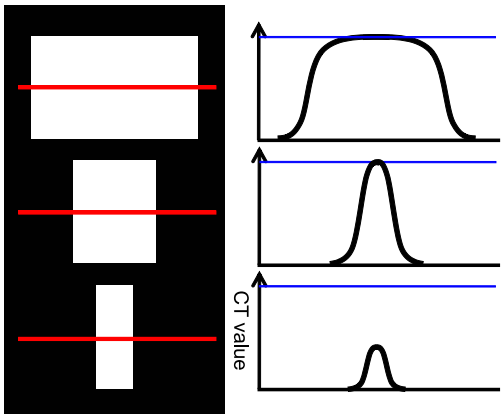


Fig. 6.2. Changes of CT values of three objects with distinct thickness.
Left: objects with distinct thicknesses. Right: graphs of CT value at the red cuts in the left figure.

On the basis of this observation, we focus on the fact that for a thin plate object, the skeletal structure can be roughly approximated by the maxima of the intensity of CT-scanned data, see Fig. 6.3.

Our aim is employing approximated intensity obtained with a spherical cover rather than raw CT-scanned values, and analytically evaluating the maxima of intensity using the discrete differential operators.

Our approximation with spherically supported functions has the following benefits:

Desired boundary. The approximation makes obtained skeletal structure reach to boundary of an object without branch, as can be seen in Fig. 6.4.

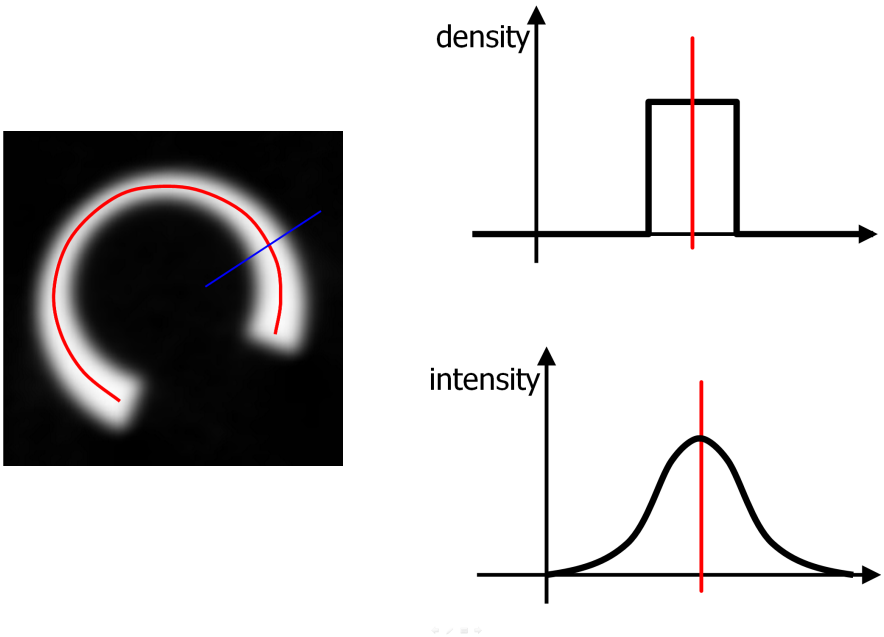


Fig. 6.3. Similarity of the skeletal structure and the maxima of intensity.
Left: a slice of CT-scanned data and its maxima (red).
Right: graph of density of an object (top) and graph of intensity of CT-scanned data (bottom) at a cut of blue line in the left image.

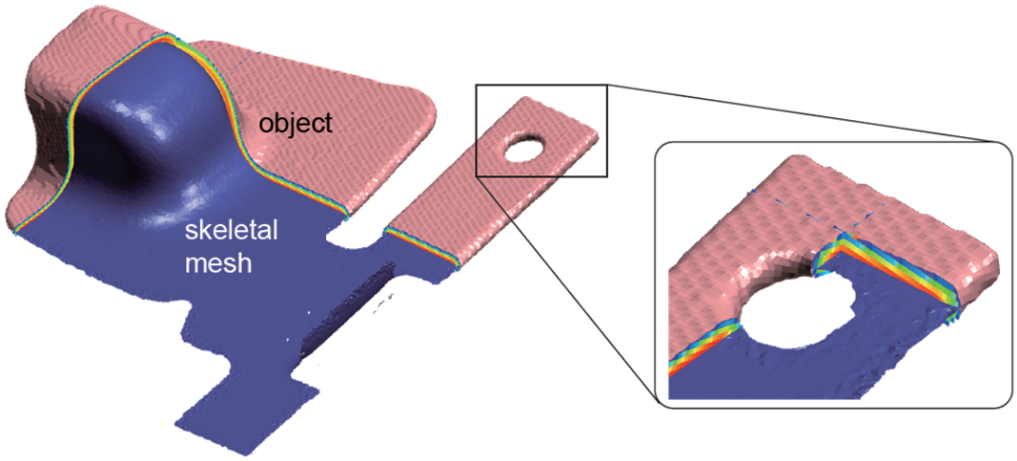


Fig. 6.4. Extracted skeletal structure reaches to the end of an object.

Adaptive smoothing. Achieving both smoothness and precise representation of a skeletal structure is a difficult task. Smooth representation can be realized with aid of generating an approximation function of intensity obtained with local approximation functions. But just a straightforward approach with a fixed support size, such as Gaussian convolution of the image [Loh98], often leads to over/undersmoothing. We arrive at the goal with adap-

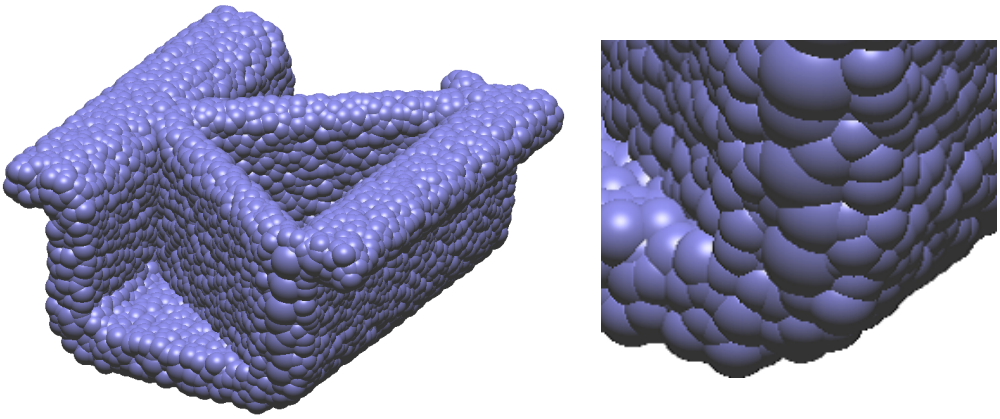


Fig. 6.5. Adaptive spherical cover and a close-up of the bottom-right corner.

tive decision of support size applying an error-driven approach. It has better smoothing effects on scanning noise while preserving important geometric features.

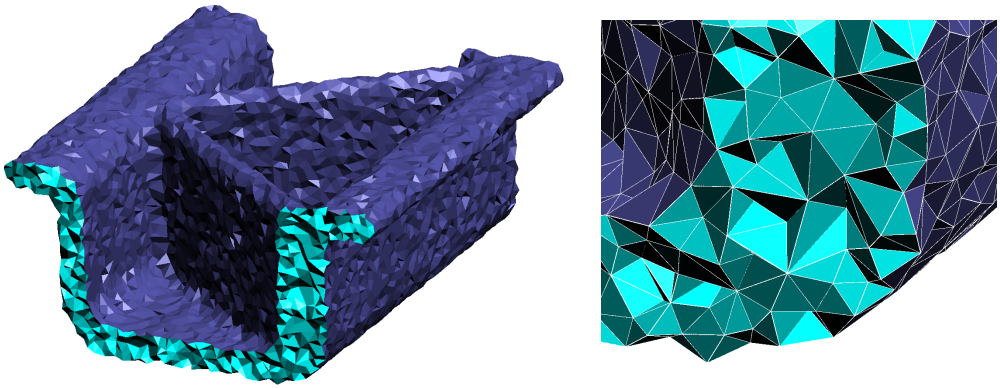


Fig. 6.6. Adaptive grid used in skeletal structure extraction and a close-up of the bottom-right corner.

Adaptive grid generation. Regularly polygonization of skeletal sheets is performed on a uniformly sampled cubical grid [FP01]. In contrast, the version used in proposed polygonization is an adaptively sized tetrahedral grid obtained from adaptive spherical supports of local approximations. Polygonization using such a grid gives adaptive sampled mesh of a skeletal structure.

Mathematical concepts

The following are more mathematical descriptions of the concepts mentioned above.

Let a 3D scalar field $f(\boldsymbol{x})$ be a C^2- continuous function defined in a domain including a given solid object. The proposed skeletal structure extraction simply finds the locus of points satisfying the following conditions:

$$\langle \boldsymbol{e}, \nabla f \rangle = 0, \quad \lambda < 0 \tag{6.1}$$

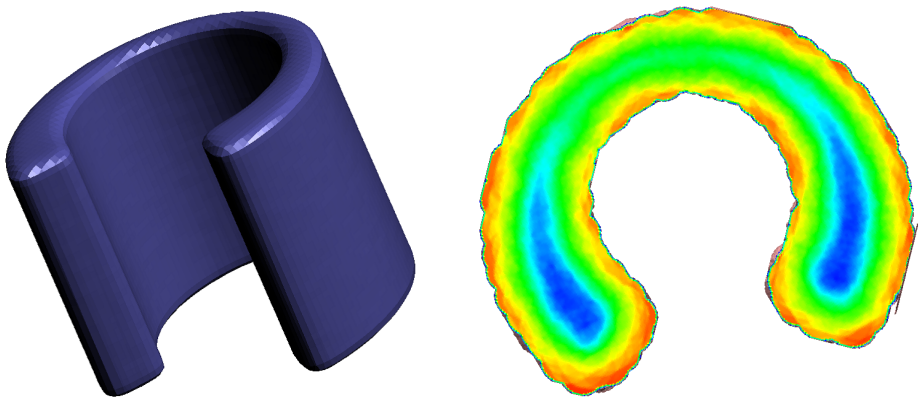


Fig. 6.7. A cross section of C^2-continuous scalar field $f(\boldsymbol{x})$.
Left: a thin object. Right: a slice of a scalar field.

where $\boldsymbol{e}$ is the eigenvector of the Hessian of $f(\boldsymbol{x})$ associated with the minimum eigenvalue λ. This condition means that $f(\boldsymbol{x})$ takes the local maximum along direction $\boldsymbol{e}$. The skeletal sheet obtained with strategy is known as a **height ridge** in image processing [Ebe96, Lin98]. In geometric modeling, surfaces defined using such extremal conditions are generally referred to as **extremal surfaces** [AK04].

A typical choice for $f(\boldsymbol{x})$ is a distance field from a solid boundary, but the skeletal sheet of a distance field is topologically too complicated: skeletal sheet has the number of branches, in the case that the boundary surface is not smooth enough. Our simple choice for $f(\boldsymbol{x})$ is an approximation of the intensity values of CT images, which are higher inside scanned objects. $f(\boldsymbol{x})$ can be constructed from CT scanned image of an object, using spherical covering technique. An example of a cross-section of a 3D scalar field can be seen in Fig. 6.7.

In order to generate a smooth approximation function of intensity, we use a spherical supports based on the partition of unity technique consisting of a set of spherically supported quadratic polynomials. This approach is similar to that in the previous chapter. But, in terms of skeletal structure extraction, only inside scalar field is enough since required skeletal structure shows the object's internal characteristics.

6.2 Algorithm

In this section, first we show the overview of the algorithm which generates a mesh approximating the skeletal sheet directly from a volume data of a thin object. Fig. 6.8 shows the overview with examples. From a volumetric data, a scalar field approximating the intensity is generated employing a spherical cover on the basis of partition of unity approach. Then the maxima of intensity which is regarded as the skeletal structure in

this algorithm is extracted as a surface mesh. The details of each step are explained in the following sections.

6.2.1 Algorithm Overview

This algorithm consists of three parts as shown in Fig. 6.8. 2D version results of each step are shown in Fig. 6.9. First, the intensity of input image is approximated through a set of adaptively spherically supported polynomials that provide an smoothed intensity field. As mentioned above, the maxima of the generated scalar field serves as a skeletal sheet. This part follows the algorithm developed in Section 3.4.2. Fig. 6.9(a) shows sampling points, and (b) is the generated spherical cover.

Next step is generation of an adaptive grid which is used in evaluation and polygonization of a skeletal structure. In this study, an adaptively sampled grid is realized by a subset of the weighted Delaunay tetrahedrization which is determined by a set of spherical supports. The generation of such a tetrahedral mesh has been explained in Section 3.5. Fig. 6.9(c) is the tetrahedral mesh generated from the spherical cover in (b).

Finally, in order to detect extremality, the derivatives are analytically evaluated on the tetrahedral mesh as shown in Fig. 6.9(d).

Algorithm: generating polygonized skeletal structure

1. Approximate the intensity of points with spherically supported functions
2. Generate an adaptive grid with a weighted Delaunay tetrahedrization
3. Find and polygonize the maxima

6.2.2 Skeletal mesh generation

In the last of the extraction algorithm, the algorithm extracts a mesh approximating the skeletal structure using the tetrahedral mesh as a grid. Results for 2D and 3D data are shown in Fig. 6.9 (d) and Fig. 6.10 respectively. In this step, crossing points with a skeletal structure and tetrahedral mesh edges are detected first. Hereafter, such a crossing point are referred as a skeletal point. For detection of skeletal points, the gradient and Hessian of the approximated intensities are used. These derivatives are estimated well with the help of the approximation function $f(\boldsymbol{x})$ provided by a spherical cover. Next, around an edge on which a skeletal point is detected, a small patch is generated, see Fig. 6.11. Vertices of each patch exist inside the tetrahedra which incident to the edge on which the skeletal point is detected. A set of patches created becomes a mesh approximating the skeletal structure.

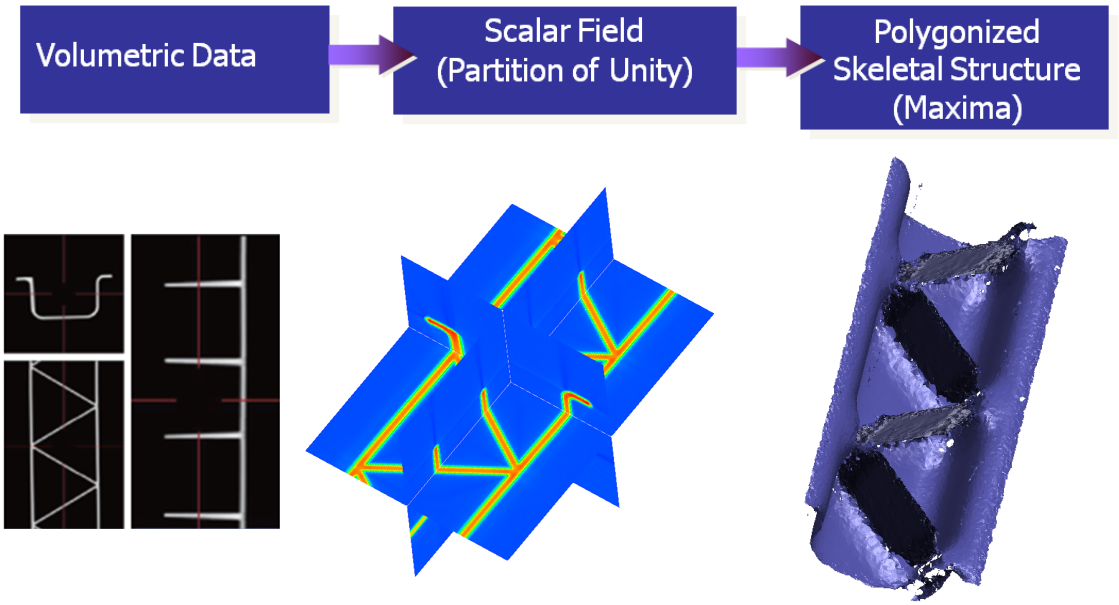


Fig. 6.8. Overview of the algorithm for skeletal structure extraction.

Skeletal points. Using a polynomial approximation function is a powerful method for estimating quality gradients and curvatures. To extract a smooth skeletal mesh, we use the gradient $\boldsymbol{g}$, the first-order derivative of the approximation function $f(\boldsymbol{x})$:

$$\boldsymbol{g} = \nabla f, \tag{6.2}$$

and the Hessian H, the second-order derivative:

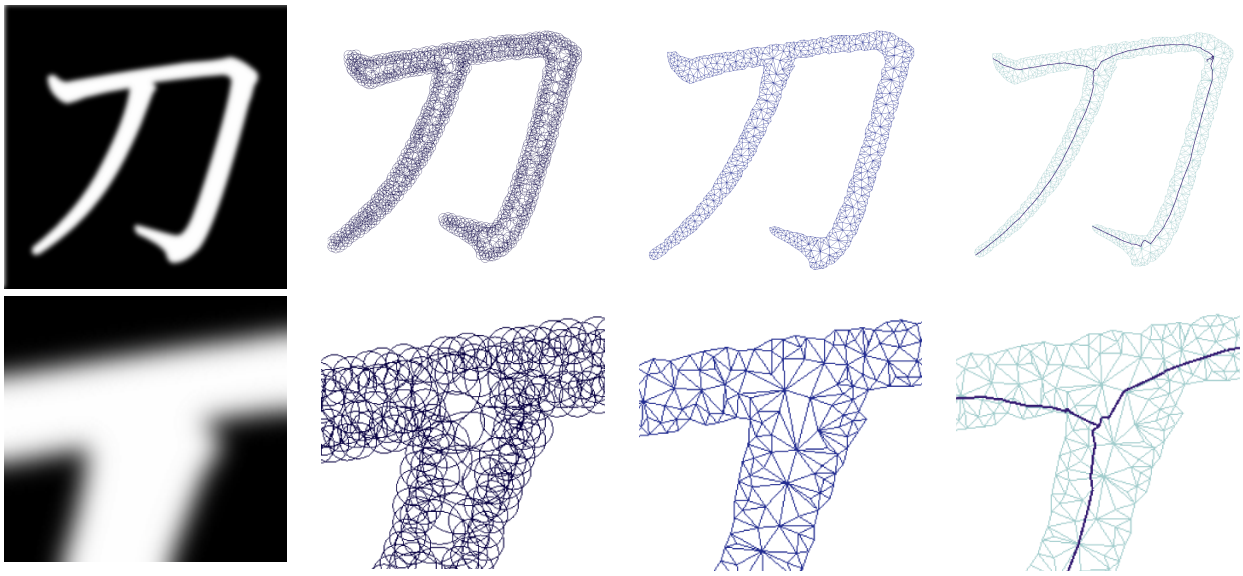
$$H = \nabla \boldsymbol{g}. \tag{6.3}$$

Let p_1 and p_2 be the end points of an edge with coordinates $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ respectively, and let the eigenvectors corresponding to the minimum eigenvalues of H at p_1 and p_2 be $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. See the image on the left of Fig. 6.11. We assume that the inner product of $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ satisfies $\langle \boldsymbol{e}_1, \boldsymbol{e}_2 \rangle > 0$. If this is not true, flip $\boldsymbol{e}_2$ to make $-\boldsymbol{e}_2$ for this condition to be satisfied.

As explained in Mathematical Concepts, the extrema are points satisfies $\langle \boldsymbol{e}, \boldsymbol{g} \rangle = 0$. Hence the existence of zero-crossing of $\langle \boldsymbol{e}, \boldsymbol{g} \rangle$ can be tested with the condition

$$\langle \boldsymbol{e}_1, \boldsymbol{g}_1 \rangle \langle \boldsymbol{e}_2, \boldsymbol{g}_2 \rangle < 0. \tag{6.4}$$

Next, for an edge that satisfies the above condition (6.4), whether the extremum is the maximum or minimum is checked. As a skeletal structure corresponds to a set of maximum points, only the crossing points corresponding to the maxima are needed. To test the maximality, the condition proposed in [OBS04] is used:



(a) (b) (c) (d)

Fig. 6.9. Overview of skeletal structure extraction in 2D.
(a) Input grayscale image.
(b) Support circles of approximation functions.
(c) Triangular mesh based on (b).
(d) Skeletal structure.

if inequation

$$\langle \boldsymbol{e}_i, \boldsymbol{g}_i \rangle \langle (\boldsymbol{x}_j - \boldsymbol{x}_i), \boldsymbol{e}_i \rangle > 0 \tag{6.5}$$

holds for $(i, j) = (1, 2)$ or $(2, 1)$, this edge has a skeletal point.
Remember the facts that $\boldsymbol{g}_i$ shows the direction of increase of $f(\boldsymbol{x})$ and that $\boldsymbol{e}_i$ shows the direction in which the changes of value of $f(\boldsymbol{x})$ is maximum. As suggested by the second fact, checking the partial differential in direction of $\boldsymbol{e}_i$ which is realized with inner product with $\boldsymbol{e}_i$ is reasonable. The above condition means intuitively that, in terms of direction $\boldsymbol{e}_i$, $\boldsymbol{g}$ increases along the edge, and thus, a maximum value exists between the endpoints. The first inner product calculates the change of $\boldsymbol{g}_i$ in the direction of $\boldsymbol{e}_i$. In the second inner product, identification of the directions of the considered edge and $\boldsymbol{e}_i$ is checked. The identify of directions of $\boldsymbol{g}$ and the edge is tested with the use of the product of these two inner products. The left image in Fig. 6.11 shows the case that detected crossing point $\boldsymbol{c}$ is a maximum point.

In the case that an edge is perpendicular to $\boldsymbol{e}_i$, this condition cannot determine existence of maximality. This is the reason that this condition is checked on the both endpoints of an edge.

Assuming that $\langle \boldsymbol{e}_i, \boldsymbol{g} \rangle$ which means the partial differential of $\boldsymbol{g}$ in direction $\boldsymbol{e}_i$ changes linearly along the edge, the coordinates of the skeletal point are calculated through interior

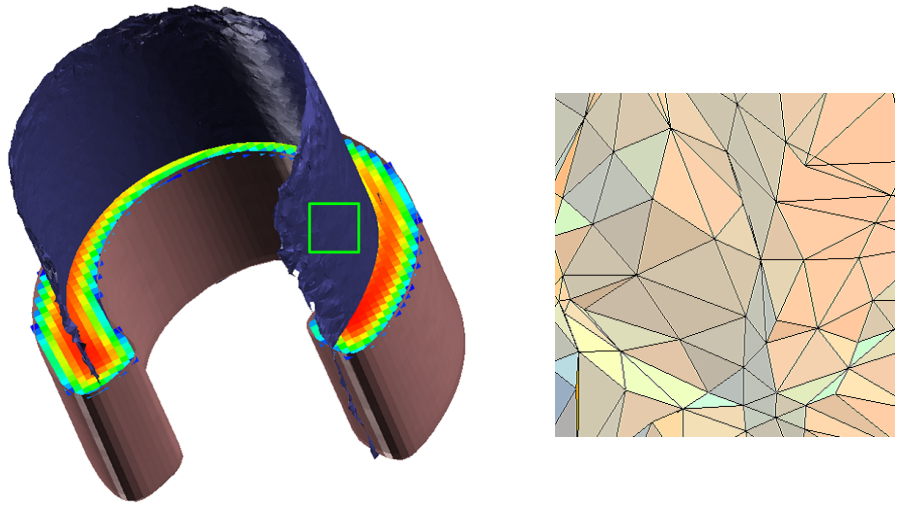


Fig. 6.10. Polygonized skeletal structure.
Left: a thin pipe-like object and mesh of its skeletal structure.
Right: wire frame of close-up of the boxed part in the left image.

division:

$$\frac{|\langle \boldsymbol{e}_2, \boldsymbol{g}_2 \rangle|}{|\langle \boldsymbol{e}_1, \boldsymbol{g}_1 \rangle| + |\langle \boldsymbol{e}_2, \boldsymbol{g}_2 \rangle|} \boldsymbol{x}_1 + \frac{|\langle \boldsymbol{e}_1, \boldsymbol{g}_1 \rangle|}{|\langle \boldsymbol{e}_1, \boldsymbol{g}_1 \rangle| + |\langle \boldsymbol{e}_2, \boldsymbol{g}_2 \rangle|} \boldsymbol{x}_2. \tag{6.6}$$

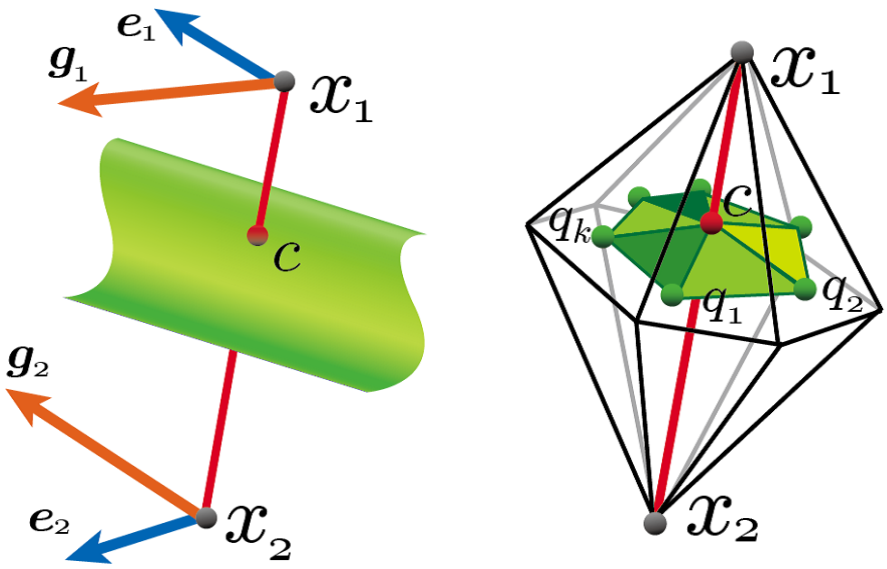


Fig. 6.11. Skeletal patch generation.
Left: edge intersecting with a skeletal structure (green sheet).
Right: generated skeletal patch around the edge.

Skeletal patches. After the detection of skeletal points, skeletal patch generation is performed. Around skeletal point c a small patch is generated as shown in the image on the right of Fig. 6.11. This patch is referred to as a skeletal patch in this thesis since the set of these patches approximates the skeletal structure. Single skeletal patch is realized with several triangles whose one vertex is c. Other vertices of a skeletal patch exist inside tetrahedra which incident to the edge on which c is detected. A vertex q_j of a skeletal patch in such a tetrahedron t_j is the centroid of all skeletal points detected on edges of t_j. All incidenting tetrahedra have such centroids since each of those has at least one skeletal point, c.

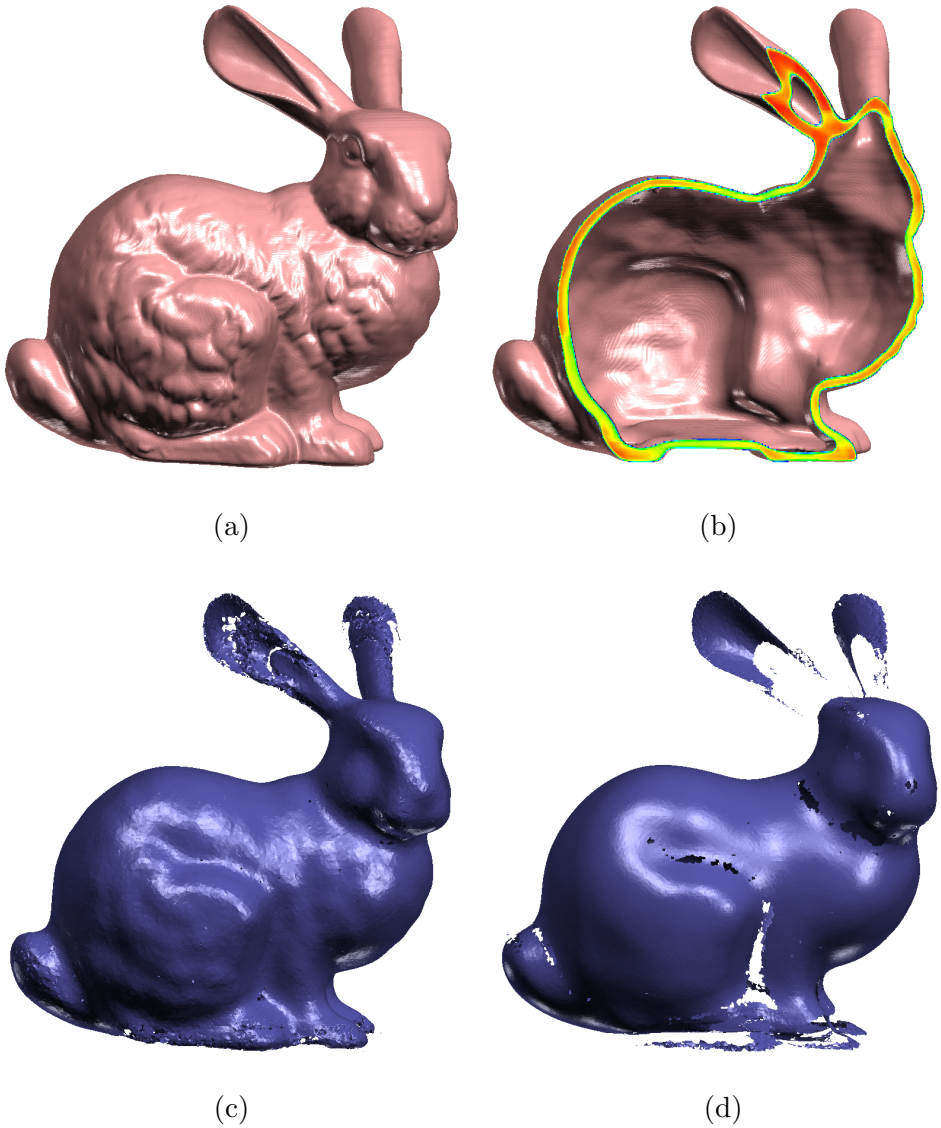
After all vertices of a skeletal patch are calculated, those vertices and c are connected with traversing all the incident tetrahedra. For instance, from a sequence of k vertices $q_1, q_2, \ldots, q_k$ a skeletal patch is generated which is realized with k triangles sharing the skeletal point c. In other words, triangles $\{q_1, q_2, c\}, \{q_2, q_3, c\}, \ldots, \{q_k, q_1, c\}$ are generated and they approximate the skeletal structure, see the right image of Fig. 6.11. This procedure is a tetrahedron-based variation of the dual contouring proposed in [JLSW02].

Smoothing with Expanding Spherical Supports. For skeletal patches to be well connected and not fragmented, it is important that the derivatives change smoothly. Such a smooth derivatives can be easily obtained through expanding the size r_i of a spherical support to $\sigma\, r_i$ ($\sigma > 1$) in the evaluation of $\boldsymbol{g}$ and H. This expansion makes the local approximation functions affect in larger regions. The all results demonstrated in this thesis are obtained with $\sigma = 4$ unless otherwise specified.

The results of this support expansion is shown in Fig. 6.12. In the result with $\sigma = 6$, the effect of smoothing is rather strong. As observed in ears, in areas where the curvature is high, extrema do not appear. Actually, in such cases the extrema exists outside the tetrahedral mesh. Since the proposed algorithm does not perform detection outside of a tetrahedral mesh, some parts of the extrema may disappear. But in such cases derivative estimation is far from accurate, and thus skeletal structure is meaningless.

6.2.3 Smoothing Local Approximation Functions

In the case that input data includes much amount of noise as CT-scanned data, the derivatives tend to be unstable. For thin objects like ones discussed in this algorithm, this adverse effect notably appears. See Fig. 6.13, the results obtained with no countermeasure against noise. The influence is especially notable around crossing-points which cannot be detected well because of the instability of derivatives. As a countermeasure against this problem, we propose to smooth the differentials of the local approximation functions using



(a) (b)

(c) (d)

Fig. 6.12. Effect of expanding spherical supports in the evaluation of derivatives.
(a) Isosurface of a CT-scanned data of the Stanford bunny.
(b) A cross-section of the scalar field defined by $f(\boldsymbol{x})$.
(c) Result with $\sigma = 3$.
(d) Result with $\sigma = 6$.

the differential operators defined in Chapter 4.

In order to improve the stability of derivations, we smooth the differentials of the local approximation functions after generating an initial spherical cover. The new algorithm for skeletal structure extraction with smoothing local functions is below.

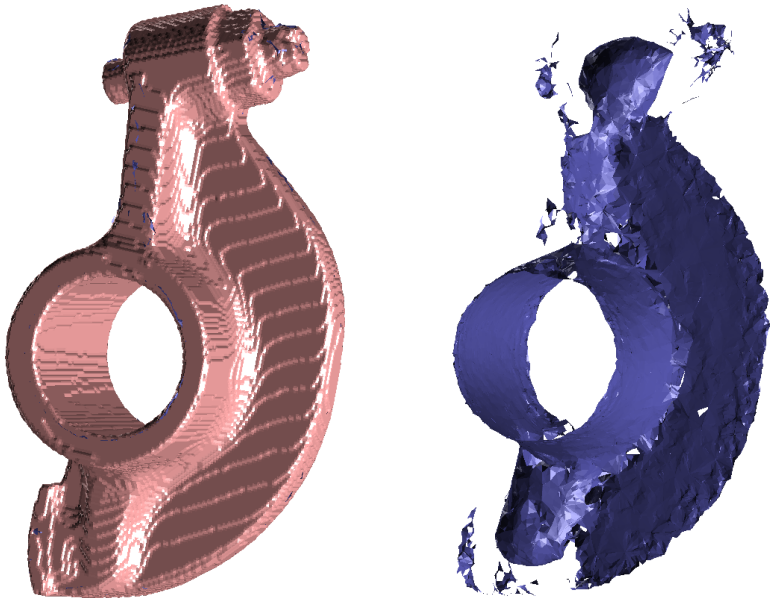


Fig. 6.13. Skeletal structure for an object with T-junction.

Algorithm: generating polygonized skeletal structure with smoothing

1. Approximate the intensity of points with spherically supported functions
2. Smooth the differentials of the local approximation functions through diffusion technique
3. Generate an adaptive grid with a weighted Delaunay tetrahedrization
4. Find and polygonize the maxima

In our skeletal structure extraction, derivative estimations are performed only on the centers of supports, and therefore we can adopt the differential operators derived in Section 5.2.3 in Step 2 of the above algorithm. In order to apply the smoothing algorithm in Section 5.2.3, there are two structures to be changed. First, a no-gap spherical cover for a region such as a bounding box which includes the all points is required. The reason is the smoothing process uses adjacency of spherical supports. But a spherical cover generated in the previous skeletal structure extraction covers only object points and points around the object. In addition, the local approximation functions $\{g_i(\boldsymbol{x})\}$ to be smoothed have to be linear. That is, new $g_i(\boldsymbol{x})$ should be a linear function

$$g_i(\boldsymbol{x}) = c_x x + c_y y + c_z z + c_0. \tag{6.7}$$

In the remarking from *uncovered* to *covered*, convex hulls determined by both kinds of points should be used.

Following these two changes, **Algorithm: generating spherical cover** has three parts that must be changed: the degree of local approximation function, the area covered

with a spherical cover, and the covering check. The new algorithm to generate a scalar field is as follows. The differences with the previous algorithm are Step 1, 2, and 3.

Algorithm: generating spherical cover

1. Initialization: set $\mathscr{P}$ as the center candidate list $\mathscr{C}$.
   Mark all points in $\mathscr{C}$ as *uncovered*
2. Spherical support generation: select a point randomly from $\mathscr{C}$ as a center $\boldsymbol{c}_i$ and decide the radius r_i and local linear function $g_i(\boldsymbol{x})$
3. Covering check: remove covered points from $\mathscr{C}$
4. Termination check: if $\mathscr{C} = \emptyset$ then the algorithm terminates.
   Otherwise go back to Step 2

Smoothing algorithm for a skeletal structure extraction is not the straight extension of the smoothing of the local approximation functions. It is because the object of our calculation is not the values of the approximation function but the values of the differentials of the approximation function. In the diffusion for surface reconstruction, we solved the Laplacian smoothing and the Poisson equation. Other equations to be solved have to be realized and formulated.

6.3 Result

Experimental results. We implemented the skeletal structure extraction except for smoothing local approximations. Some results for CT scan data are shown in Figs. 6.14, 6.15 and 6.16. The coloring on the cross section shows the input CT values (red for high and blue for low). Each of extracted skeletal structure extends to the ends of an object without branching. Fig. 6.14 gives the results for objects with uniform and nonuniform thickness. Proposed algorithm works well for both examples. The object in Fig. 6.16 has a noisy boundary. This algorithm can extract a smooth skeletal structure even from such data. The bumps on the surface of the original object do not affect the results.

Comparison with image processing approaches. Since input data is a CT image, it is also possible to extract skeletal voxels using conventional image processing approaches. Given a certain value and a direction at each voxel, non-maximum suppression [Can86] is one of the most popular method for finding the voxels taking local maximum value along the directions. For extracting skeletal voxels, we use the standard Gaussian convolution of CT values to obtain the value and direction at each voxel [Loh98].

As shown in the top images of Fig.6.17, the image processing approach works well for the

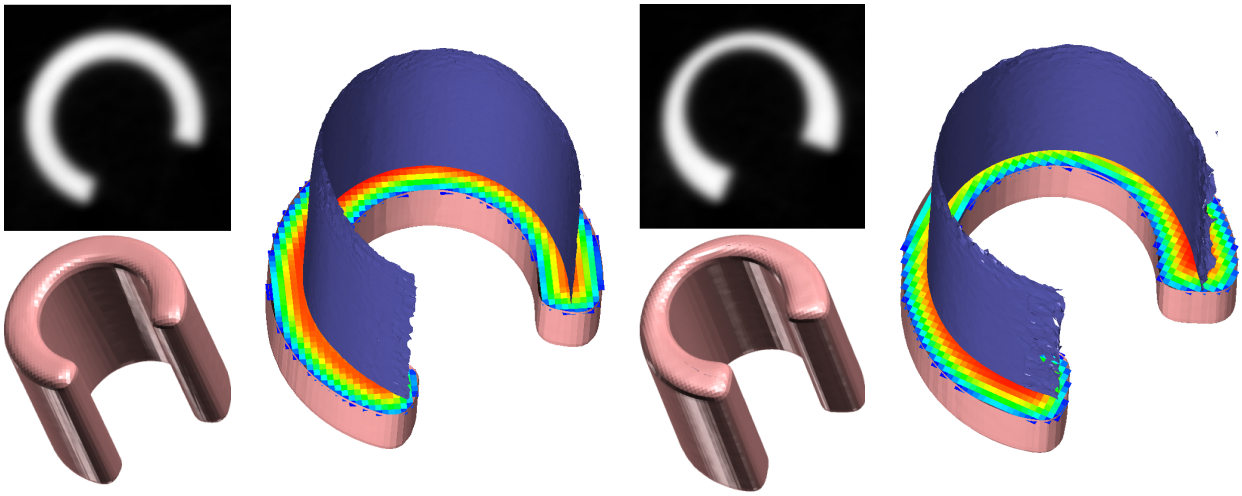


Fig. 6.14. Skeletal structures of objects with uniform and non-uniform thickness.
Left: results for a CT scanned object with a uniform thickness.
Right: results for a CT scanned object with a non-uniform thickness.

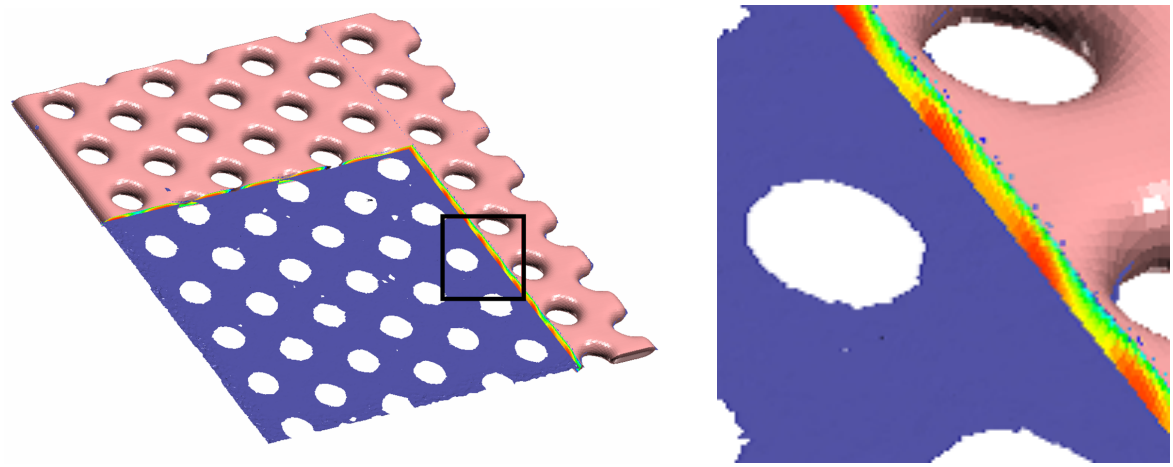


Fig. 6.15. Skeletal structure for a perforated metal plate.

model with a uniform thickness if we choose a proper kernel size of Gaussian convolution. However, it is hard to obtain a good result for the model with a non-uniform thickness because the fixed kernel size causes under-smoothing (fail to eliminate unwanted branches) or over-smoothing (skeletal voxels go out of the object). See Fig.6.18 for examples of such problems. Parts of voxels are not extracted properly. Result in the left figure has branches because of under-smoothing, and the right has disappeared part because of over-smoothing. In contrast, proposed method automatically adjusts the support sizes of local approximations applying the error analysis in equation (3.11). Thus a proper smoothing effect can be adaptively obtained.

Moreover, to generate a mesh approximating a skeletal structure from extracted voxels, we have to apply a surface reconstruction method to connect the centers of extracted

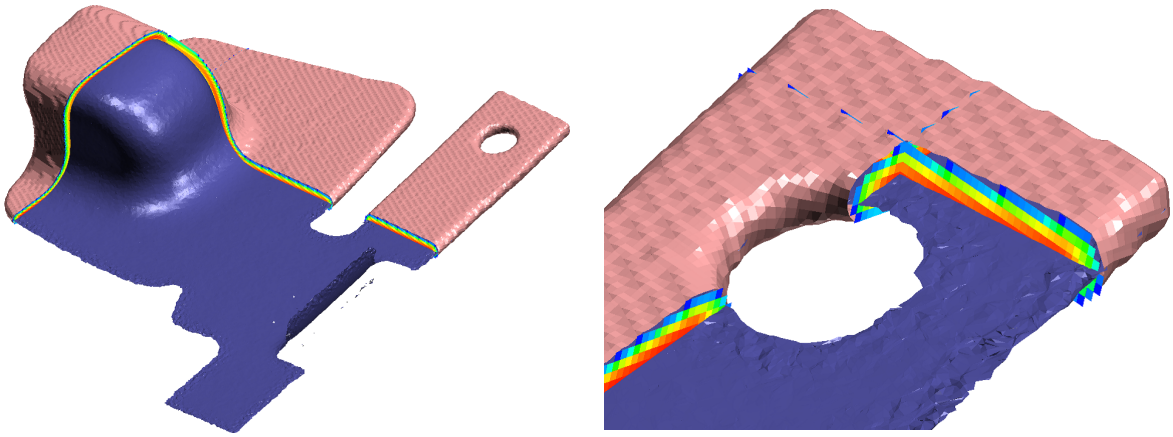


Fig. 6.16. Skeletal structure for a thin plate with a hole.

voxels with polygonal patches. The bottom images of Fig. 6.17 show triangular meshes obtained from the skeletal voxels. Since the mesh vertices are located on the grid points, the meshes are not smooth enough unlike the meshes obtained through proposed method. Further, some of surface reconstruction methods are fail to reconstruct meshes because the grid artifact behaves as large noise.

Compared with the Gaussian convolution of a 3D image, proposed algorithm is an expensive task. For example, it takes about ten minutes on a standard PC for computing the result shown in Fig. 6.16 while about three minutes for Fig. 6.17. This is the only drawback of proposed method against the non-maximum suppression with the Gaussian convolution. However, proposed method does not require a grid sampling structure to the input volume. Therefore it is possible to apply this method to scattered volume samples (irregularly sampled points with values) which cannot be handled by image processing approaches.

6.4 Discussion

Instability in derivative calculation. Theoretically speaking, the algorithms proposed in Section 6.2.2 can handle even T-junctions as long as edges which should have skeletal points are detected. But actually we observed that the derivatives around such parts are very unstable, and thus we found detection for non-manifold parts is practically a difficult problem. See the image (b) in Fig. 6.19. As shown in surface reconstruction in Section 5.2.3, smoothing the local approximation functions as proposed in Section 6.2.3 will greatly help improvement of stability. We plan to implement this proposal and compare its results with current results.

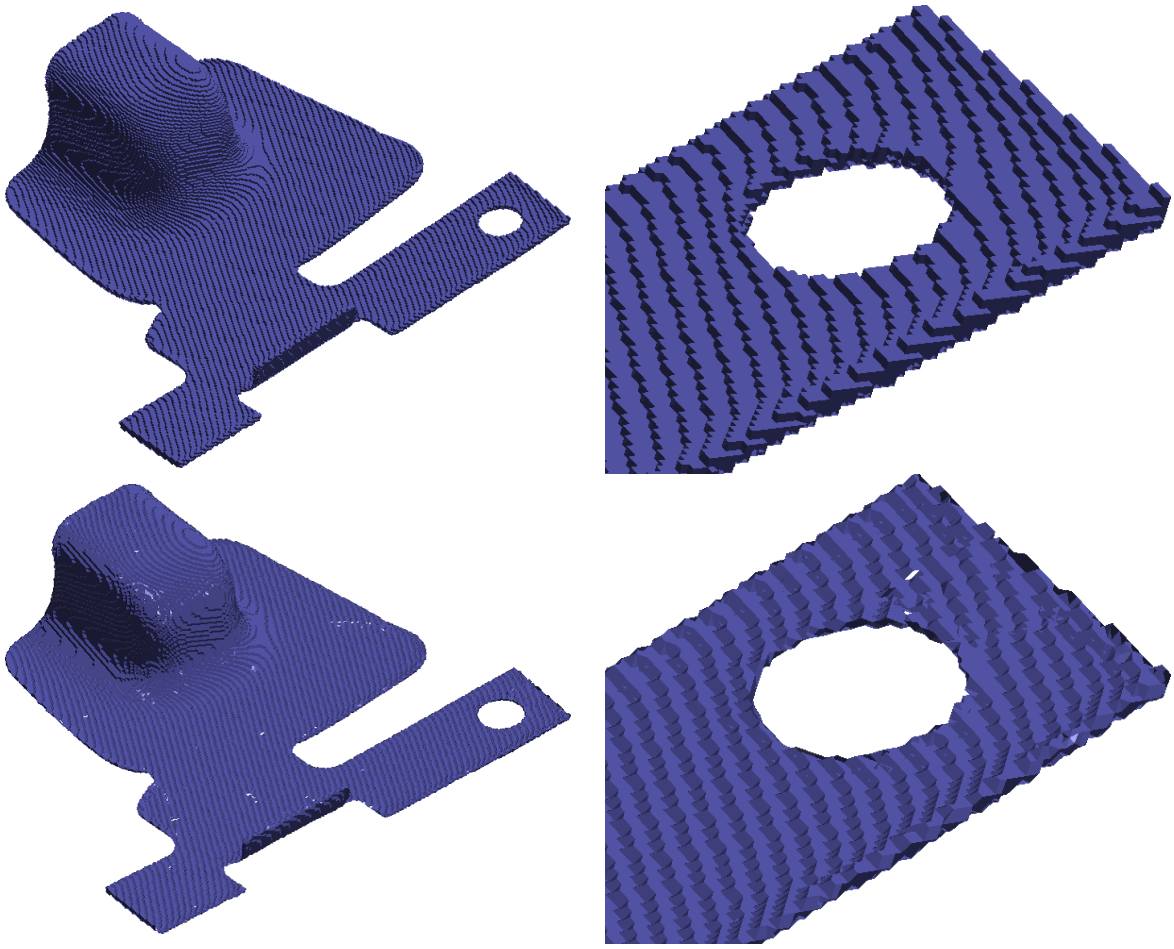


Fig. 6.17. Results obtained using non-maximum suppression for a CT image in Fig. 6.16.
Top: extracted skeletal voxels.
Bottom: a triangular mesh generated connecting the skeletal voxels.

Cleaning. The result meshes in this chapter are *not* cleaned at all. So the extracted skeletal structures may have a few unwanted fragments, see Fig. 6.19(c). To discard these fragments and obtain only a skeletal structure, cleaning up generated mesh using certain restrictions is required. For instance, weighting area of patches employing the absolute values of the eigenvalues of the Hessian and thresholding the value may help to delete fragments.

Existing tiny holes as in the image (d) of Fig. 6.19 is also a problem. The reason of appearance of those holes is the degeneracy of a tetrahedral mesh. Changing the perturbation into theoretically more concrete one will eliminate such degenerate parts. Symbolic perturbation is an instance of theoretically guaranteed perturbation.

Decision of support size. Another remaining subject is deciding the minimum and maximum radii of support spheres. The support size should play an important role in this algorithm, but the relationship lies outside the analysis of this thesis. We would like to

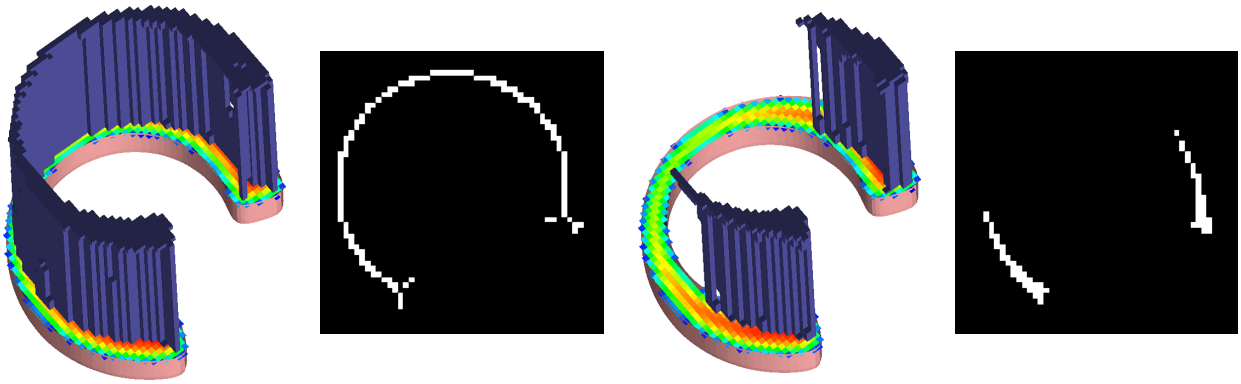
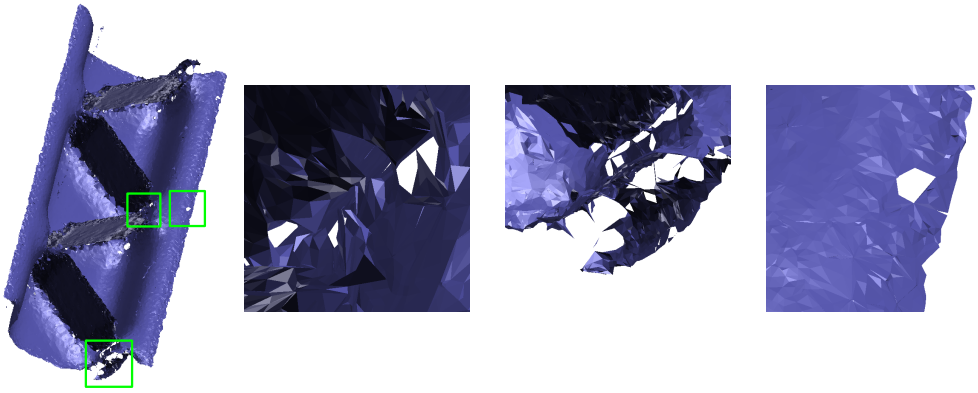


Fig. 6.18. Skeletal voxels obtained using non-maximum suppression for the right model in Fig.6.14.
The original object is a model with a non-uniform thickness.



(a) (b) (c) (d)

Fig. 6.19. Problems in generated mesh and close-ups of them.
(a) Generated mesh with problems.
(b) Area failed to be extracted.
(c) Undesired fragments.
(d) Holes generated in polygonization.

make progress toward decision of these values.

Timing. As mentioned in the last part of Section 6.3, the proposed algorithm takes high computational cost. Especially tetrahedrization takes a large part of the computational time. The tetrahedrization explained in this chapter is a straightforward way for a spherical cover. However, it is an $O(n^5)$ algorithm for n the number of supports. Adopting another grid structure can reduce the computational time although Delaunay-based tetrahedrization has an advantage of generating an adaptive grid.

Formulation for smoothing. In general, volumetric data includes much amount of noise. So smoothing before evaluating the maxima is considered as a good solution. But the object of this smoothing is not the values of the approximation function. This is the difference

between the smoothing for surface reconstruction. Formulating of the smoothing for skeletal structure extraction is a future work.

6.5 Summary

In this chapter, an algorithm to extract a triangular mesh approximating the sheet-like skeletal structure of a CT-scanned thin object is proposed. This algorithm consists of approximation based on a spherical covering technique and derivation on an adaptive grid. Using proposed algorithm, a smooth mesh can be directly obtained from scanned data. Because of the approximation strategy, proposed algorithm can detect a smooth skeletal structure that reaches almost to the ends of the object.

Chapter 7

Conclusion and Future Work

We extended a spherical covering method with deriving discrete differential operators for a spherical cover with linear local approximation functions. Equipping proper differential operators expands the applications of spherically supported functions. As instances of applications using spherically supported functions, we also developed two algorithms in surface reconstruction and skeletal structure extraction. In this chapter, we summarize the contributions of this thesis first, next describe future research directions.

7.1 Summary of Contributions

Differential Operators on a Spherical Cover. We derived discrete differential operators on centers of spherical supports: gradient, divergence, and Laplacian. In the derivation, we assumed two conditions: assignment of a constant vector to each spherical support and discretization of boundary of a support. In many cases these assumptions are reasonable and proposed operators can be directly introduced.

Proposed operators are proper to a spherical cover and considered suitable for processing on a spherical cover. Thus using proposed operators is expected to provide better results than ones with traditional continuous operators.

Because of their fundamental nature, proposed operators can be applied to various kinds of attributed data such as volumetric data. We also developed two application algorithms for point cloud from a surface of an object and volumetric data respectively. In these algorithms, differential operators are used to reduce influence of noise included in input data.

Surface Reconstruction. The goal of surface reconstruction is precise and smooth representation.

We arrived at the former property on the basis of partition of unity approach with a spherical cover. The drawback of this method is dependency on the quality of data; the method is especially sensitive to problematic parts in scanned data such as noise,

outliers, lack of sampling, and differences in sampling density. These problems make surface reconstruction an essentially difficult subject.

We achieved the robustness for outliers, lack of sampling, and differences in sampling density through combining Graph-cut to the partition of unity approach. Difficulties caused by noise, lack of sampling, and differences in sampling density are overcome with diffusing local approximation functions of a spherical cover. The diffusion is performed on a spherical cover employing Laplacian smoothing and solving Poisson equation, in both operations the discrete differential operators are used. Smooth representation is realized these two strategies.

In addition, we proposed a novel measuring of confidence of sampling points. This measurement is defined through observation of the difference of angles between normals of PCA plane and sampling point. This measurement can be directly performed on a spherical cover and detect outliers effectively.

Skeletal Structure Extraction. We developed an algorithm to generate a surface mesh approximating skeletal structure directly from volumetric data of a thin-plate and one-material structure. This algorithm is developed through an observation that the skeletal structure is very close to the maxima of intensity of volumetric data.

First the algorithm generates a scalar field approximating the intensity of volumetric data employing a spherical cover, and then the skeletal structure is extracted with evaluating derivatives. Using approximated intensity makes evaluation of derivatives stable, as a result, algorithm becomes robust to noise.

Generated mesh has no branch in its end unlike results generated employing strategies employing medial-axis. In addition, ends of obtained mesh extend to the boundary of an object. This character is useful for some applications.

7.2 Future Directions

Relaxation of assumptions in derivation of differential operators. In this thesis, to derive the discrete differential operators, we made following two assumptions for a spherical support:

- a constant vector is assigned (Assumption 4.3.1),
- a part of boundary covered with another support is not included the other supports. (Assumption 4.3.2).

These are strong simplifications and sometimes the artifacts caused by the unreality of these assumptions are observed in the tiny oscillations on reconstructed surfaces, see Fig. 7.1. Tiny bumps appear in the foot which is expected to be smooth.

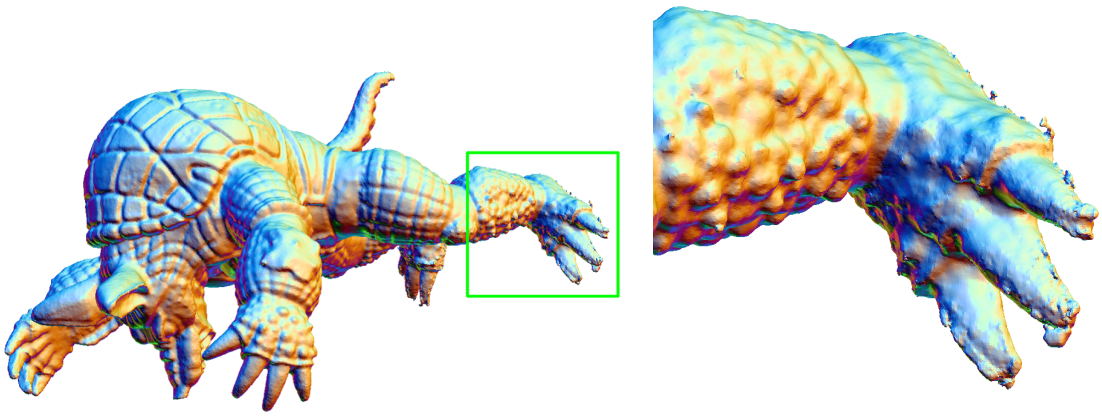


Fig. 7.1. Artifacts caused by the unreality of assumptions in deriving the differential operators.

Only in terms of cases such as our two applications which use vector fields defined with the gradients provided approximation functions with sufficiently small errors, Assumption 4.3.1 is practical. In such cases, the difference of vectors at arbitrary two points inside a support is not so large, and thus this assumption is not so hard in practice.

For the second assumption, although a normalized term is introduced in equation (4.9) to negate the influence of this assumption, the actuality of this assumption may require more verifications and experiments.

Derivation of other differential operators. Although we derived the gradient, divergence, and Laplacian operators on a vector field, and the Laplacian operator on a scalar field, the curl on a vector field is not refined in this thesis. Curl plays also important roles in applications, and thus, deriving it will represent an advance in its application area. We plan to achieve it, for instance, following an energy calculation performed in [TLHD03].

Derivation of Discrete Integration Operators. In this study discrete differential operators have been derived. Since the integrations also play key roles in many geometrical processing as differential operators do, derivation of discretized integration operators is another worthwhile subject. It will make possible direct calculations of fundamental quantities as area of surfaces and volume of objects through no conversions of shapes into meshes. Spherically supported functions with discrete integration operators will become more powerful and excellent representation.

Diffusion of higher order local approximation functions. The linearity assumption is natural if the finally required representation is mesh. It is verified with experiments demonstrated in this thesis that the ability of representation is enough high to satisfy a required level in

most applications. However, it is also true that raising the order of local approximations to quadratic or cubic equations broadens the area of application.

The diffusion technique in Chapter 5 is developed completely on the basis of the linearity of local approximations: the vector field is defined with the gradients and Poisson equation is derived from the discrete divergence operation on the gradient of approximation function. That is, the reason of the dependency on the linearity is the assumption of discrete operators. Thus, for changing the order of local approximations, changing the derivation of the discrete operators with more sophisticated one is required.

Diffusion of local approximation functions with changing support sizes and center positions. The support size and its center are decided with error-driven approach in the first step of the algorithm, and it is not changed in the following steps. Changing the support size and the position of the center after the updating of local approximations may improve the accuracy of reconstructed model, and thus this direction warrants study.

Adaptive Smoothing. The smoothing proposed in this thesis smoothes all the local approximations uniformity. The size of the supports reflects the details of the surface inside, so changing the level of the smoothing adaptively is expected to be a better detail-preserved smoothing.

Reconstruction of sharp features. For surface reconstruction, we proposed an adaptive smoothing method mainly employing smoothing technique. This algorithm can roughly reserve edges, but cannot represent sharp features. Reservation of sharp features is an important subject as smooth representation, so it is an important subject. One straightforward solution is assigning more than one local approximation as MPU [OBA+03]. Grouping input points inside a support with the direction of normals, and then each group is fitted a local approximation. Finally the inside/outside of an object is recognized through set operations for the local approximations.

1D and 2D skeletal structure extraction. This is a direction for generating skeletal structures with higher ability of representation. It is worth extracting skeletal structures composed to line-like structures and sheet-like structures. Such mixture of 1D and 2D skeletal structures are useful for more intuitive understanding object shapes. The proposed algorithm can become a fundamental method of extraction of such structures.

Processing Higher Dimensional Data. Nowadays many attributed scanned data can be available. We believe that the proposed smoothing strategies are suitable for such data. Possible applications are, for instance, approximation of intensity values of time-varying CT-scanned data (4D to 1D), and reconstruction with colors (6D to 4D, transforming

spacial points with colors to the signed distance with color attributes). These are very attractive subjects because of the expectations that extra attributes make approximations more stable or precise ones.

Acknowledgment

I wish to extend my gratitude to my supervisor Professor Hiromasa Suzuki for his overall supports over the three years. He gave me many fruitful advices in basis on his advanced knowledge and experiments. He also gave sincere encouragements for me to pursue my studies. I would like to express my appreciation to Professor Yutaka Ohtake for his highly specialist advices in computer graphics. His aid for implementations also greatly improved the quality of this research. In addition, MPU executable, viewers for various types of geometric structures, and risu and Siisa-A models are by courtesy of Professor Yutaka Ohtake. I would like to express my gratitude to Professor Kokichi Sugihara for his comprehensive knowledge. He was also the supervisor in my master's degree and taught me the way a researcher should be. I am grateful for Professor Akira Iwasaki's fruitful discussions. He gave me many comments which brush up this research. Professor Takashi Kanai gave advices which improved the quality of this thesis a lot. His comments made several words in this thesis more sophisticated terms. I would like to thank Doctor Kiwamu Kase, the team leader of the VCAD modeling team in RIKEN, and the team members for having given me a fruitful experience to research together. I thank all the members in Suzuki Laboratory and Ohtake Laboratory. I want to thank professors and officers of the department of advanced interdisciplinary studies, the school of engineering, the University of Tokyo. Finally I would like to thank to my family who supported me every time.

This research is partly funded by the Tokyo University AIS transcosmos Special Scholarships and Exchange and the school of engineering the University of Tokyo.

The Stanford bunny, the Stanford dragon, the Happy Buddha are provided by the Stanford Computer Graphics Laboratory. The executables of Poisson surface reconstruction and Graph-cut are publicly available thanks to Michael Kazhdan, and Yuri Boykov and Kolmogorov respectively.

Bibliography

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, 2001.

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of ACM SIGGRAPH 1998*, pages 415–421, 1998.

[ACK01] N. Amenta, S. Choi, and R. Kolluri. The power crust. In *Proceedings of the 6th ACM Symposium on Solid Modeling*, pages 249–260, 2001.

[ACSTD07] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-based variational reconstruction of unoriented point sets. In *SGP '07: Proceedings of the 5th Eurographics symposium on Geometry processing*, pages 39–48, 2007.

[AK04] N. Amenta and Y. Kil. Defining point-set surfaces. *ACM Transactions on Graphics*, 23(3):264–270, 2004. Proceedings of ACM SIGGRAPH 2004.

[BK04] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[Blo88] J. Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, 1988.

[Boi84] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, 1984.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

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

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of ACM SIGGRAPH 1996*, pages 303–312,

1996.

[CSM07] N. D. Cornea, D. Silver, and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization And Computer Graphics*, 13(3):530–548, 2007.

[CSYB05] N. Cornea, D. Silver, X. Yuan, and R. Balasubramanian. Computing hierarchical curve-skeletons of 3d objects. *The Visual Computer*, 21(11):945–955, 2005.

[CT03] P. Choudhury and J. Tumblin. The trilateral filter for high contrast images and meshes. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 186–196, 2003.

[DG03] T. K. Dey and S. Goswami. Tight cocone: A water-tight surface reconstructor. In *Proc. 8th ACM Sympos. Solid Modeling Applications*, pages 127–134, 2003.

[DG06] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. *Computational Geometry: Theory and Applications*, 35(1):124–141, 2006.

[DMSB99] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999.

[DZ03] T. K. Dey and W. Zhao. Approximating the medial axis from the voronoi diagram with a convergence guarantee. *Algorithmica*, 38:179–100, 2003.

[Ebe96] D. H. Eberly. *Ridges in Image and Data Analysis*. Kluwer Academic, Dordrecht, Nederlands, 1996.

[Ede93] H. Edelsbrunner. The union of balls and its dual shape. In *Proceedings of the 9th annual symposium on Computational geometry*, pages 218–231, San Diego, California, United States, 1993.

[FKS06] T. Fujimori, Y. Kobayashi, and H. Suzuki. Separated medial surface extraction from ct data of machine parts. *Lecture Notes in Computer Science*, 4077:313–324, 2006. Geometric Modeling and Processing - GMP 2006.

[FP01] J. D. Furst and S. M. Pozer. Marching ridges. In *Proceedings of the IASTED International Conference on Signal and Image Processing*, pages 22–26, 2001.

[FSKK05] T. Fujimori, H. Suzuki, Y. Kobayashi, and K. Kase. Contouring medial surface of thin plate structure using local marching cubes. *ASME Journal of Computing and Information Science in Engineering*, 5(2):111–115, 2005.

[GG07] G. Guennebaud and M. Gross. Algebraic point set surfaces. In *SIGGRAPH*

'07: ACM SIGGRAPH 2007 papers, pages 23.1–23.9, 2007.

[GH95] A. Guéziec and R. Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):328–342, 1995.

[Gib98] S. F. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 23–30, 1998.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of SIGGRAPH 1992*, pages 71–78, 1992.

[HK06] A. Hornung and L. Kobbelt. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *SGP '06: Proceedings of the 4th Eurographics symposium on Geometry processing*, pages 41–50, 2006.

[JBC07] T. Ju, M. Baker, and W. Chiu. Computing a family of skeletons of volumetric models for shape description. *Computer-Aided Design (accepted)*, 2007.

[JLSW02] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, 2002.

[KBH06] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *SGP '06: Proceedings of the 4th Eurographics symposium on Geometry processing*, pages 61–70, 2006.

[KCVS98] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114, 1998.

[KSO04] R. Kolluri, J. R. Shewchuk, and J. F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 11–21, 2004.

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

[Lin98] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):117–156, 1998.

[Loh98] G. Lohmann. *Volumetric Image Analysis*. Wiley, 1998.

[MWO03] W.-C. Ma, F.-C. Wu, and M. Ouhyoung. Skeleton extraction of 3d objects with radial basis functions. In *Shape Modeling International 2003*, pages 207–215, 2003.

[NOS09] Y. Nagai, Y. Ohtake, and H. Suzuki. Smoothing of partition of unity implicit surfaces for noise robust surface reconstruction. *Computer Graphics Forum*, 28(5):1339–1348, 2009. Proceedings of Symposium on Geometry Processing 2009.

[OBA+03] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level partition of unity implicits. *ACM Transaction of Graphics*, 22(3):463–470, 2003.

[OBA05] Y. Ohtake, A. Belyaev, and M. Alexa. Sparse low-degree implicit surfaces with applications to high quality rendering, feature extraction, and smoothing. In *Proceedings of the 3rd Eurographics symposium on Geometry processing*, pages 149–158. Eurographics Association, Aire-la-Ville, Switzerland, 2005.

[OBS02] Y. Ohtake, A. Belyaev, and H.-P. Seidel. Mesh smoothing by adaptive and anisotropic gaussian filter. In *Proceedings of Vision, Modeling, and Visualization VMV 2002*, pages 203–210, 2002.

[OBS03] Y. Ohtake, A. G. Belyaev, and H.-P. Seidel. Multi-scale and adaptive CS-RBFs for shape reconstruction from cloud of points. In *MINGLE workshop on Multiresolution in Geometric Modelling*, pages 337–348, Cambridge, UK, 2003.

[OBS04] Y. Ohtake, A. Belyaev, and H.-P. Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on Graphics*, 23(3):609–612, 2004. Proceedings of ACM SIGGRAPH 2004.

[ÖGG09] C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009. Proceedings of EUROGRAPHICS 2009.

[PH02] S. Prohaska and H.-C. Hege. Fast visualization of plane-like structures in voxel data. In *Proceedings of the conference on Visualization*, pages 29–36, 2002.

[PSM94] P. Perona, T. Shiota, and J. Malik. Anisotropic diffusion. In *Geometry-Driven Diffusion in Computer Vision*, pages 73–92, 1994.

[PT90] B. A. Payne and A. Toga. Surface mapping brain function on 3d models. *Computer Graphics and Applications, IEEE*, 10(5):33–41, 1990.

[PTVF93] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

[Ren88] R. J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software*, 14(2):139–148, 1988.

[Set99] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999. Second Edition.

[SLS+07] A. Sharf, T. Lewiner, G. Shklarski, S. Toledo, and D. Cohen-Or. Interactive topology-aware surface reconstruction. *ACM Transaction of Graphics*, 26(3):43.1–43.9, 2007.

[Tau95] G. Taubin. A signal processing approach to fair surface design. In *Computer Graphics (Proceedings of SIGGRAPH 95)*, pages 351–358, 1995.

[Tau01] G. Taubin. Linear anisotropic mesh filters. Technical Report RC-22213 (W0110-051), IBM Research Technical Report, 2001.

[TLHD03] Y. Tong, S. Lombeyda, A. N. Hirani, and M. Desbrun. Discrete multiscale vector field decomposition. *ACM Transactions on Graphics*, 22(3):445–452, 2003.

[TWBO02] T. Tasdizen, R. Whitaker, P. Burchard, and S. Osher. Geometric surface processing via normal maps. Technical Report UUCS-02-003, University of Utah School of Computing, 2002.

[WK04] J. Wu and L. P. Kobbelt. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum*, 23(3):643–652, 2004. Proceedings of EUROGRAPHICS 2000.

[WRC88] G. J. Ward, F. M. Rubinstein, and R. D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92, New York, NY, USA, 1988. ACM.

Appendix A

Graph-cut

Graph-cut often appears in image processing because of its advantage to divide areas with target attributes even from an image with ambiguous boundaries. Graph-cut is equal to solving a problem known as Max-flow min-cut problem in graph theory.

A **graph** G is a structure which is composed to nodes V and edges E which are lines connect subset of nodes. In the case edges are assigned numerical weights, the graph is called as a **weighted graph**.

Let us consider a weighted graph with nonnegative weights. A **cut** $C = (S, T)$ is a partition of nodes into two disjoint groups S and T. Edges whose end nodes are belong to different groups are called as **cut-set**. The **weight** of cut is defined as the sum of weights of cut-set. **Graph-cut** is a problem to find the cut with the minimum weight. In Graph-cut, two special nodes *source* and *sink* are assigned which are referred as **terminal nodes**. Graph-cut finds the minimum cut $C = (S, T)$ which source $\in S$ and sink $\in T$.

Fig. A.1 shows an example of extraction of an area from an image. Usually, each pixel has a corresponding node although only several nodes are represented in these figures. The left image shows a weighted graph and a target area (pink region). The center image is the result obtained with Graph-cut. The set of yellow edges is the cut-set. Nodes are colored as the groups which belong to. As a result, target area can be obtained through extraction of the souse-side nodes.

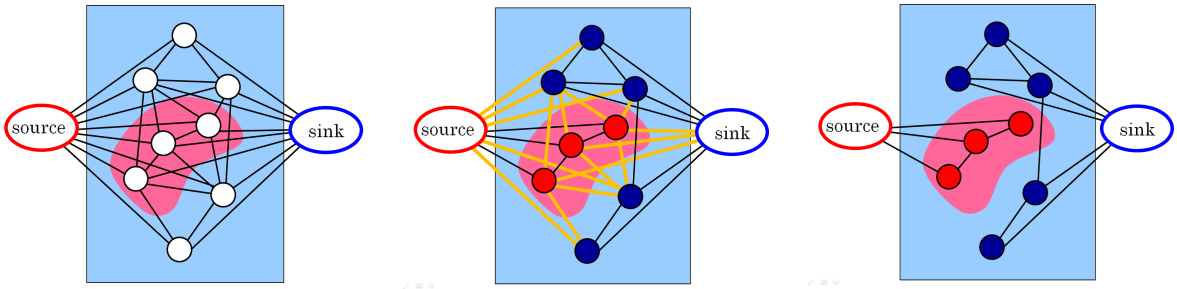


Fig. A.1. Graph-cut in 2D.
Left: source and sink nodes are determined. Center: the minimum cut is obtained through Graph-cut. Right: the cut classifies nodes into two groups.

Appendix B

Speed-up of Calculating Connectivity of Support Spheres

In the algorithms introduced in this thesis, many operations require accessing neighbors of spheres. Calculating neighbors every time they are needed is very time consuming. For achieving fast computation, we construct a kd-tree of centers of support spheres and a list of neighbors of each support sphere. We refer these structures while processing following operations.

B.1 Data Structure

A list of neighbors of spherical supports is referred to as a **sphere intersection graph** (**SIG**). SIG is realized by a list whose each element is a list of neighbors of a spherical support. After the generation of a spherical cover, we construct a kd-tree of centers and SIG. For a support sphere s_i with the center $\boldsymbol{c}_i$ and the radius r_i, centers inside a sphere whose center and radius are c_i and $2r_i$ are collected. And then a support sphere s_j corresponding to a collected center is checked whether intersections with s_i. If s_j has an intersection, s_i and s_j are neighbors of each other. Note that the radius of s_j is small, s_i may not be recognized in the term of s_j. But in the term of s_i, s_j can be recognized. So a pair of intersecting supports s_i and s_j is detected, add each supports into the lists of neighbors of the others.

B.2 Queries

While processing the proposed algorithms, two kinds of geometrical queries are made frequently. One is for collecting points inside a certain sphere and the other is for collecting spheres including a certain point. In order to accelerate these two geometrical queries, we use two other data structures. The former query can be achieved with a kd-tree of points. Starting with the nearest point from the center of specified support, checking whether a

point which is the next of the included point is within the specified sphere is enough. For collecting spheres including a specified point, we use an octree-based structure introduced in [WRC88]. The basic idea is storing large spheres in large octants and small spheres in small octants. We recommend to refer the article for details.

B.3 Performance

In order to improve the time and memory for calculation, reducing the number of degree of SIG is reasonable. The most simple way is shrinking the support sizes while obtaining the connectivity of SIG. Shrinking $0.6 \approx 0.8$ times of the original sizes is recommended. More sophisticated way is restricting the maximum number of the degree using priority queue. It can be ordered by the area of intersection, for instance. Note that this restriction may occur some artifacts in your results.

Appendix C

Polygonization of Iso-surface

The polygonization of an isosurface of a scalar field or volumetric data has been studied well. The Marching Cubes [LC87] is the early one and followed by many variants and improvements because of its simplicity and good results. The variants are the dual contouring [JLSW02] and the marching tetrahedra [PT90, GH95] which we changed for the Graph-cut in Section 5.2.2, for instances. The basic idea of these polygonization algorithms is as follows, this polygonizes the isosurface with zero value. See also the Fig. C.1 which shows the above algorithm in 2D.

1. Make a grid in the area including the isosurface if the data is not volumetric data (volumetric data has axis-aligned points)
2. Evaluate the values on all grid points of each grid cell
3. Make a surface inside each grid cell which has at least one edge with different signs of values

In step3, vertices of the surface are on edges with different signs of values, as the right image of Fig. C.1. The positions of vertices are determined on the edge through interpolation with the absolute values of the assigned values of the end points. Fig. C.1 shows the case polygonizing the zero-level surface (yellow line) between the area inside the object (positive value, pink) and the area outside (negative value, blue). The obtained polygon is shown with green lines and points. In a 3D case, grid cells are cubes while in 2D they are rectangles.

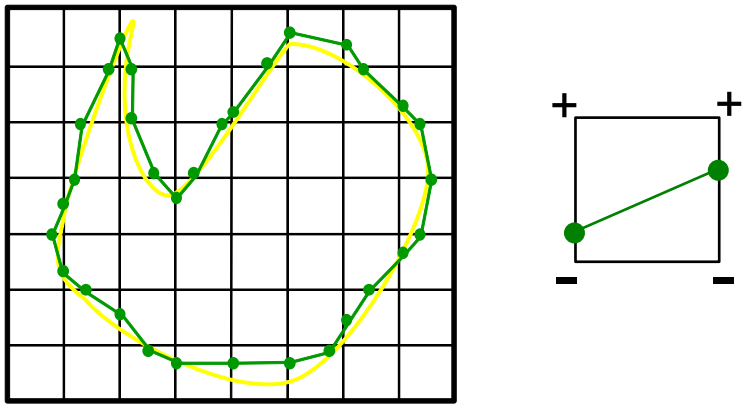


Fig. C.1. The Marching cubes in 2D.

Index

List of Publications

Refereed Journal Publications

1. Yukie Nagai, Yutaka Ohtake and Hiromasa Suzuki. Smoothing of Partition of Unity Implicit Surfaces for Noise Robust Surface Reconstruction. *Computer Graphics Forum*, 28(5):1339-1348, 2009. Proceedings of Symposium on Geometry Processing 2009.
2. 長井超慧, 杉原厚吉, 鈴木宏正, 頂点追加による 2 次元 DistMesh の等方性向上, 日本応用数理学会 論文誌, 日本応用数理学会, Vol. 17, No. 3, pp.195-217, 2007.
3. 長井超慧, 杉原厚吉, 滑らかな境界を表現する等方的四面体メッシュ生成法, 日本応用数理学会 論文誌, 日本応用数理学会, Vol. 17, No. 3, pp.347-361, 2007.

Refereed Conference Proceedings

1. Yukie Nagai, Yutaka Ohtake and Hiromasa Suzuki. Noise Robust Surface Reconstruction by Combining PU and Graph-cut. In *Proceedings of Eurographics 2009*, pages 73-76, short paper, 2009.
2. Yukie Nagai, Yutaka Ohtake, Kiwamu Kase and Hiromasa Suzuki. Extraction of Skeletal Meshes from Volumetric Data by Sparse Polynomial Approximation. In *CGA '08: Proceedings of the eighth Annual International Workshop on Computational Geometry and Applications*, pages 413-420, 2008.
3. Yukie Nagai, Yutaka Ohtake, Kiwamu Kase and Hiromasa Suzuki. Polygonizing Skeletal Sheets of CT-Scanned Objects by Partitioin of Unity Approximations. In *SMI'08: Proceedings of Shape Modeling International*, pages 265-266, poster, 2008.
4. 長井超慧, 大竹豊, 鈴木宏正, グラフカットによる異常値耐性のある陰関数曲面生成法, Outlier Robust Implicit Surface Reconstruction Using Graph-cut, Visual Computing/グラフィクスと CAD 合同シンポジウム, 2009, Visual Computing/グラフィクスと CAD 合同シンポジウム 2009 予稿集, 09-11, 2009 年. グラフィクスと CAD 研究会優秀研究発表賞 受賞.

Talks

1. 長井超慧, 大竹豊, 鈴木宏正, 滑らかさを考慮した陰関数曲面生成, Smoothing Implicit Surfaces for Partition of Unity, 2009 年度精密工学会秋季大会，2009 年 9 月.
2. 長井超慧, 大竹豊, 鈴木宏正, グラフカットを用いた点群からのノイズ耐性のある陰関数曲面生成, Noise Robust Surface Reconstruction using Implicit Function with Graph-cut, 2009 年度精密工学会春季大会, 2009 年 3 月.
3. 長井超慧, 大竹豊, 鈴木宏正, 適合的な多項式近似を用いた薄板の CT データに対する中立面抽出, Skeletal Structure Extraction from CT Scanned Thin-Object via Adaptive Polynomial Approximation, 2008 年度精密工学会春季大会, 2008 年 3 月.
4. 長井超慧, 大竹豊, 鈴木宏正, 多項式近似を用いた薄板のボリュームデータからのスケルトン構造抽出, 日本応用数理学会 2008 年研究部会連合発表会, 2008 年 3 月.

Unpublished Manuscripts

1. 長井超慧, 大竹豊, 鈴木宏正, 薄板のボリュームデータのための多項式近似によるスケルトンメッシュ生成, 情報処理学会 論文誌, 情報処理学会, Vol.51, No.3 掲載予定.