DATA-DRIVEN 3D SHAPE MODELING

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

YONGJIAN XI

Dr. Ye Duan, Dissertation Supervisor

MAY 2010

The undersigned, appointed by the dean of the Graduate School, have examined the dissertation entitled

DATA-DRIVEN 3D SHAPE MODELING

presented by Yongjian Xi,

a candidate for the degree of Doctor of Philosophy,

and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Ye Duan

Professor Dong Xu

Professor Jianlin Cheng

Professor Gang Yao

ACKNOWLEDGEMENTS

I would first and foremost like to express my deep gratitude to my adviser, Dr. Ye Duan, for his guidance, inspiration and support to me during years.

Special thanks are due to Dr. Dong Xu, Dr. Jianlin Cheng and Gang Yao, for their kindness in serving on my thesis committee.

I’d also like to give thanks to my colleagues, Gregory Curtis Heckenberg, Qing He and Kevin Karsch, for their assistance during my research.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Today, computer is used extensively to manipulate 3D model in computer vision, computer animation and visualization. 3D shape modeling is a very fundamental and essential issue and is bridging the gap between computers in the virtual world and human beings in the real world. Computers cannot understand the objects and shapes in the real world thus will require shape modeling techniques to extract the 3D information of real world objects from source data, or raw data, such as photos, points, and volumetric data.

Currently, there exists some software users can use to create 3D models in computer. However the 3D models created by these software are not good enough. For example, in medical imaging, we have to reconstruct 3D model from the patient's real data, such as MRI, CT.

Today, people are producing more and more raw data, thanks to the technology breakthrough. Those numerous data can be categorized into three major types, volumetric data, points cloud and multi-view images. Volumetric data is in fact, a 3D array, which samples discrete points in 3D space by a uniformly sampling rate, and it can be generated by ultrasound, CT, MRI and lots of other data acquisition equipments. Points cloud is obtained by laser or other type scanners, which contains discrete, but not uniformly sampling points in 3D space. The multi-view images are just 2D images, taken by one or several cameras, at different positions. Now the digital camera is so popular, even the very low end digital camera can produce very high quality and high resolution photos. This type of data is the most easily obtained but the most difficult to be processed and translated into 3D model.

In the virtual world, computer can only understand limited information coded with specified data structures. Unfortunately, the numerous types of data mentioned above cannot be processed directly when computer tries to manipulate the underlying 3D object. Of course, computer is capable of processing such raw data. E.g., image processing for 2D photos, or complex calculation on volumetric data. But before the raw data is converted to model understandable by computer, they can hardly be used for manipulating in 3D space (virtual world).

At this point, 3D shape modeling is essential for computer to understand our real world. So far, 3D shaping modeling is still an open issue. There are too much raw data around, but there is no uniform or standard way to translate them for computers. My work is an attempt to build a bridge from real world to virtual world.

We proposed several algorithms to process those three major types of raw data, and try to translate them into triangulation mesh, which can be understood by almost all of applications manipulating 3D object. The first algorithm is our region growing surface extraction algorithm, which reconstructs underlying objects from volumetric data and generates high quality triangle meshes. For points cloud data, we propose a nonparametric noisy points preprocessing algorithm, which eliminates the noise points. The cleaned points cloud can be further transformed to volumetric data for 3D reconstruction. The multi-view stereo is the most challenging, which reconstructs the 3D object by several photos. We proposed iterative surface evolution algorithm, which combines deformation model and our previous two algorithms for volumetric data and points cloud.

In 3D shape modeling or reconstruction, it's very important to transform one type of dataset to another. Reconstruction from volumetric data has been studied for years; if we can transform points cloud or 2D images into volumetric data, then we can use existed algorithms to process those data. To unify the 3D shaping modeling processing, data transformation is the key component; it acts like a hub to connect different types of dataset. And even more, if there are new sorts of dataset emerging with the new technology developed, we can transform such dataset into the dataset we've already known well and use existed algorithm. This idea is deeply within our algorithms.

The following part is organized as:

Chapter 2 proposes our region-growing iso-surface algorithm based on Marching Triangles, and targets volumetric dataset.

Chapter 3 introduces nonparametric noisy points preprocessing, it filters out noise from points cloud dataset and makes it easy be transformed to volumetric dataset for further processing.

Chapter 4 introduces my work on multi-view stereo, which combining deformation model and the algorithm on chapter 2 and 3.

Chapter 5 is summary, includes conclusion and future work.

2 Region-Growing Iso-Surface Extraction

2.1. Introduction

Implicit surfaces defined by volumetric data have become very popular in computer graphics, computer vision, geometric modeling, and computer animation, etc. Their advantages are numerous, ranging from ease of use to compact storage and powerful shape blending operations. They are particularly convenient for modeling smooth objects of arbitrary topology and geometry such as objects with holes, branches, and handles. Nonetheless, for many applications, an explicit polygonal representation of the implicit surface, defined by the zero-level iso-surface of a three-dimensional scalar field is still necessary and preferred. On the other hand, recent technical breakthroughs in new imaging modalities such as CT, MRI and Ultrasound as well as other 3-D scanning technologies have given rise to massive volumetric datasets. How to extract and reconstruct the shape of 3-D objects from these datasets accurately and efficiently remains to be both extremely challenging and significant in computer graphics, medical imaging, and visualization, etc. In a nutshell, a good iso-surface extraction algorithm should meet the following three criteria: (1) *Geometrically accurate*, i.e. the extracted polygonal mesh should be as close to the original shape as possible and preserves small features; (2) *Topologically correct*, i.e. the polygonal mesh is homeomorphic to the original shape; and (3) *Representation efficient*, i.e. the sampling rate of the polygonal mesh is adaptive to the local geometric properties (such as curvature) and maintains a good triangle aspect ratio thus no small thin triangles will be generated.

To date, there have been a lot of methods proposed for extracting 3D surface from volumetric datasets. They can be classified into two main categories: volumetric-based cell decomposition method and surface-based region growing method. The most popular volumetric-based approach is the Marching Cubes algorithm proposed by Lorenson and Cline [lorensen87] in 1987. In their algorithm, a cube is bounded by eight voxels located on two adjacent slices. Each grid point is coded as either inside or outside the object w.r.t. the surface-defining threshold. Based on the configuration of vertices that lie inside and outside the object, the cube is triangulated. Because marching cubes algorithm generates at least one triangle per voxel through which the surface passes, this may result in an enormous number of extremely small or thin triangles. Postprocessing algorithms such as mesh optimization [hoppe93] or mesh simplification [heck97] are often needed to improve the mesh quality and reduce the mesh size.

Over the years, many variants of the Marching Cubes algorithm have been published [nielson03, ouh05]. To resolve ambiguities for cube types, Montani *et al.* [montani94a] proposed a lookup table that prevents holes by consistently separating positive vertices on ambiguous faces. Nielson *et al.* [nielson91] made use of bilinear contour topology to resolve ambiguities on boundary faces. Cignoni *et al.* [cign00] extended this concept by examining the trilinear interpolation in a cell's interior and completely determining piecewise trilinear isosurface topology. Another powerful method to resolve ambiguity is adaptive subdivision, such as the octree technique. In principle, subdivision-based, adaptive techniques also improve the accuracy and efficiency of the surface approximation through the use of spatial partitioning [wilh92, wyvi88]. Kobbelt *et al.* [kobbelt01] presented an Extended Marching Cube algorithm for feature-sensitive

surface extraction from volumetric data using directed distance fields, in order to reduce the alias effect. Gibson *et al.* [gibs98] proposed another type of volumetric-based algorithm ”Dual Methods" that generates one vertex lying on or near the contour for each cube that intersects the contour. For each edge in the grid that exhibits a sign change, the vertices associated with the four cubes that contain the edge are joined to form a quad. Topologically, the mesh generated by the dual methods is the dual of the mesh generated by the Marching Cubes algorithm. Since the vertices of the mesh can move within the cube instead of being restricted to the edges of the grid as in Marching Cubes algorithm, the mesh quality is generally better. Recently, several new "Dual Methods" algorithms [ju02, ohta02d, nielson04, warr04] have been proposed that can either represent sharp feature [ju02, ohta02d]}, generate smoother meshes [nielson04], or extract small thin features adaptively without excessive subdivision of the underlying volumetric grid [warr04].

While most of existing iso-surface extraction algorithms are based on cell decomposition, Hilton *et al*. proposed a surface-based region growing algorithm - the Marching Triangles algorithm [hilt96]. Comparing with the aforementioned volumetric-based algorithms, the Marching Triangles algorithm can generate much higher quality evenly-sized and quasi-equilateral meshes by iteratively joining new triangles to the current region boundary. The original Marching Triangles algorithm however does not adapt well to local surface properties and thus requiring large numbers of small triangles to approximate surfaces with large variations in curvature. Recently, researchers have proposed several curvature-adaptive iso-surface extraction algorithms

[akko01, kark01, rodr04] based on the concept of the Marching Triangles algorithm [hilt96].

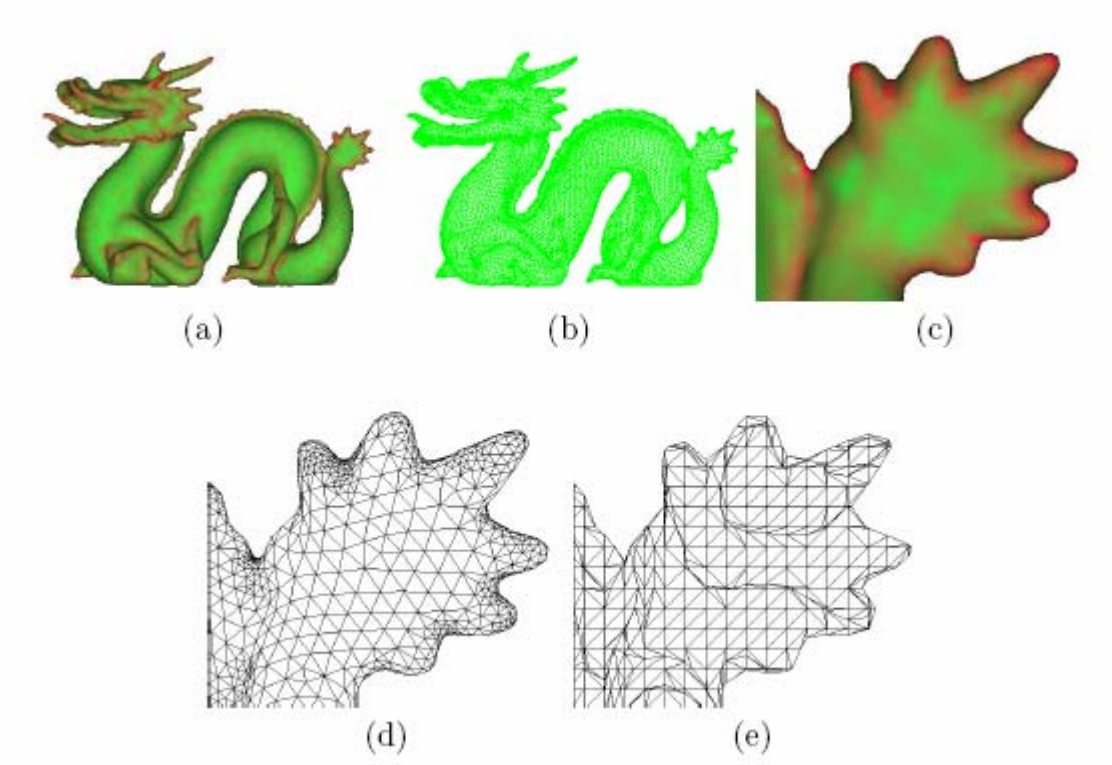


Figure 1 : Semi-regular curvature adaptive iso-surface extraction

(a) Curvature map of the volumetric dragon; (b) extracted mesh; (c) close-up view of the curvature map; (d) close-up view of the corresponding mesh; (e) close-up view of mesh generated by Marching Cubes.

We propose a new region-growing based iso-surface extraction algorithm that is also based on the concept of the Marching Triangles algorithm. The main contribution is that we propose a novel normal consistency constraint that ensures the intersection of the Delaunay sphere of the new triangle and the iso-surface is a topological disk, an important property which is lacking in the original Marching Triangles algorithm [hilt96] and its existing variants [akko01, kark01, rodr04]. In addition, by employing a prioritized

seed initialization scheme, the new algorithm can generate meshes (e.g. Figure 1) that are not only curvature-adaptive, but also semi-regular, which maybe more preferable for most applications. Moreover, it will extract all the disjoint components of the iso-surface, and can preserve sharp features.

2.2. Algorithm

The entire pipeline of the algorithm (Figure 2) consists of the following five main steps:

1. Boundary Particles Sampling and Classification.
2. Feature Positions Extraction.
3. Prioritized Seed Initialization.
4. Front Propagation.
5. Visited Boundary Particles Identification.

After the first two preprocessing steps (Step 1 and Step 2), the algorithm will loop from Step 3 to Step 5 until all the iso-surfaces are extracted.

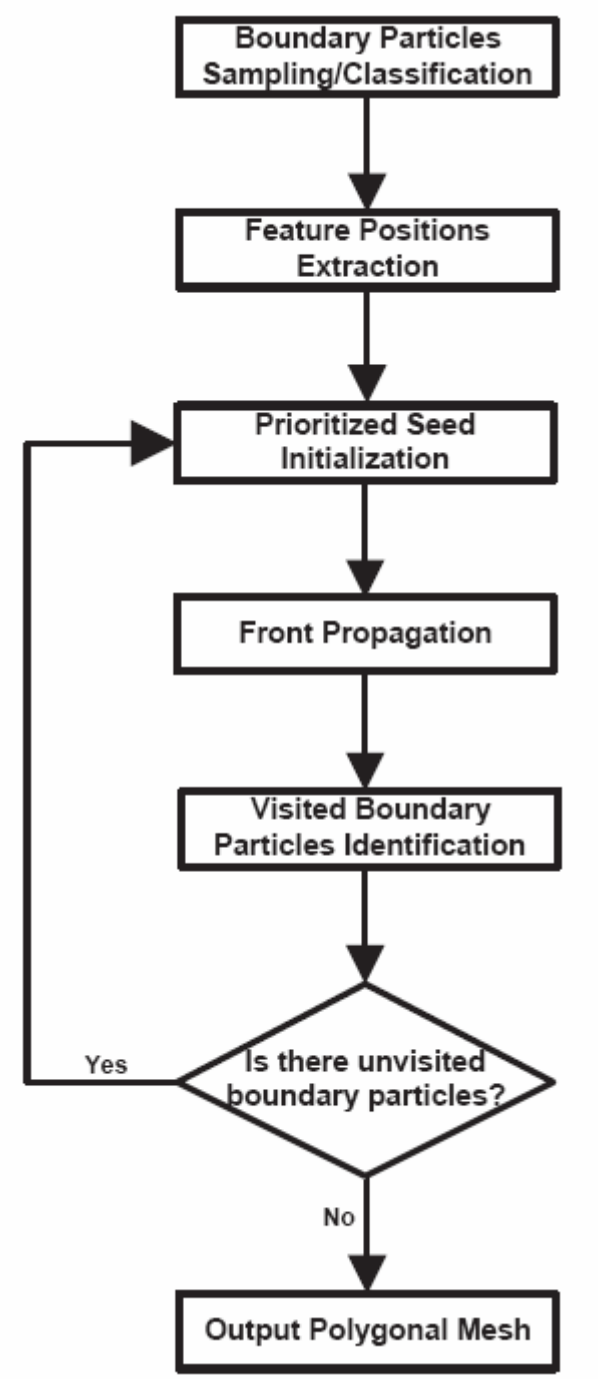


Figure 2 : Algorithm flow chart

2.2.1. Boundary Particles Sampling and Classification

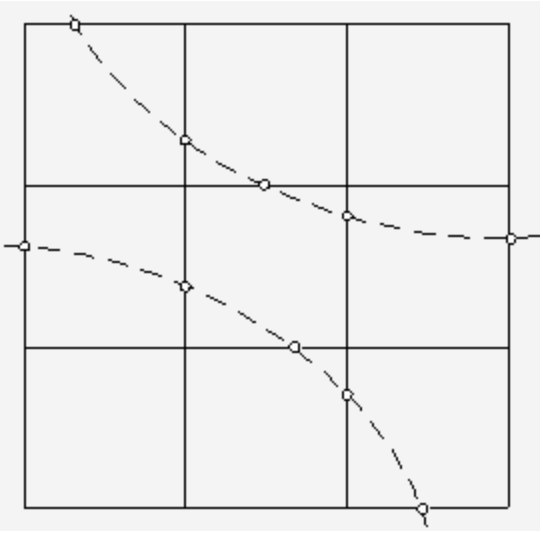


Figure 3 : Boundary particles sampling

Boundary particles (empty circles) are sampled at the intersections of the iso-surface (dashed lines) and the edges of the boundary cells (solid lines). They will then be classified according to their local geometric properties (e.g. curvature) and will be used for seed initialization.

The first step of the algorithm is boundary particles sampling. Figure 3 shows a 2D illustration. Boundary particles (empty circles) are sampled at the intersections of the iso-surface (dashed lines) and the edges (solid lines) of the boundary cells (i.e. cells that intersect the iso-surface). These boundary particles will then be further classified into *n* classes according to their local geometric properties such as curvature (we use the maximum absolute principle curvature in this thesis). To estimate these geometric properties such as gradient and curvature, instead of using the commonly used trilinear interpolation, we use a more accurate cubic interpolation filter as suggested in [moller98]. The classified boundary particles will then be inserted into the seeding priority list and will be used for seed initialization. The whole algorithm will stop only after all the

boundary particles are visited by one of the propagating front (Section 2.2.5). This will ensure all the disjoint components of the iso-surface will be extracted.

2.2.2. Feature Positions Extraction

Our algorithm can preserve sharp features. For example, if Hermite type of datasets (i.e. datasets with exact intersection points and normals) are available, we can extract the sharp feature points/edges by solving the Quadratic Error Function (QEF) as suggested in [kobbelt01] and [ju02]. The extracted feature positions will then be inserted with the highest priority into the seeding priority list and will be used for seed initialization.

2.2.3. Prioritized Seed Initialization

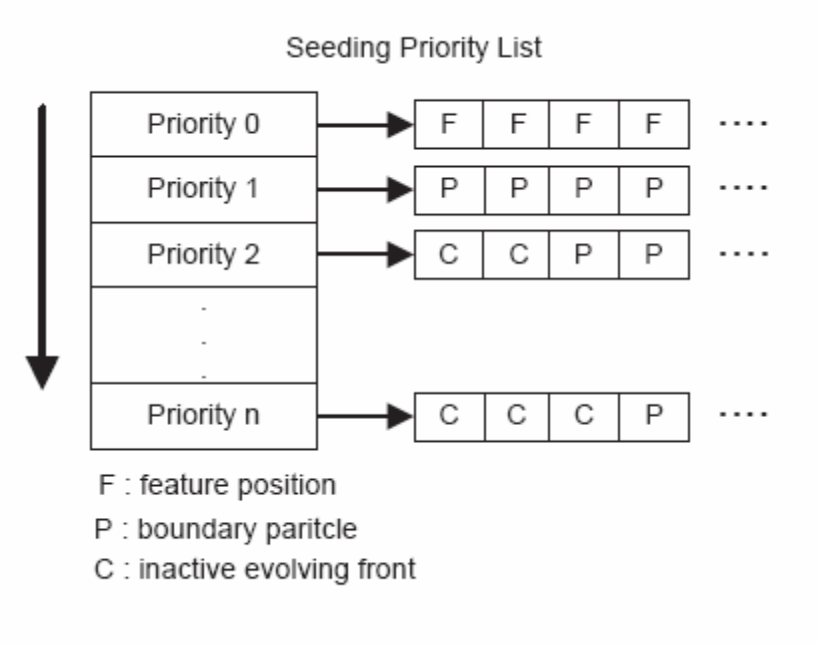


Figure 4 : Prioritized seed initialization.

Candidate positions for seed initialization are organized into a priority list. Extracted feature positions are assigned the highest priority and are put in the top row (Priority 0) of the priority list. Boundary particles and pointers of all inactive evolving fronts are put into the next *n* rows (Priority 1 to Priority n) of the priority list according to the classifications.

Seed initialization will be conducted in a hierarchical fashion by placing all the candidate seeding positions into a priority list. The seeding priority list can be implemented as a bucket linked list (Figure 4 : Prioritized seed initialization.). Extracted feature positions are assigned the highest priority and are put in the top row (Priority 0) of the priority list. Boundary particles are put into the next *n* rows (Priority 1 to Priority n) of the priority list according to their geometric properties such as curvature. We use the maximum absolute principle curvature to classify the boundary particles. A suggested

step size is associated with each row in the priority list with the largest step size in the top two rows and decreasing step sizes (e.g. one half of the previous step size) in the following *n-1* rows. In general, the higher curvature the boundary particle has the lower priority it will be and the smaller step size it will be assigned to. In our implementation, we initialize the priority list with three rows: the top row (Priority 0) stores all the extracted features; the second row (Priority 1) stores all the boundary particles whose curvature is below the average curvature of all the boundary particles; the third row (Priority 2) stores all the boundary particles whose curvature is greater or equal to the average curvature of all the boundary particles. The top row and the second row will start with the original step size, while the third row will have one half of the original step size. When a propagating front seeded from the Priority *k* row of the priority list becomes inactive, it will be freezed and the pointer of the front will be inserted into the beginning of the Priority *k+1* row of the priority list. This way, when this front is later reactivated, it will evolve at a step size smaller than its current step size (e.g. one half).

This prioritized seed initialization scheme will make the propagating front grows in a layer-by-layer fashion so that at each layer the majority of the triangles will be about equilateral. Hence our algorithm can generate meshes not only curvature adaptive, but also semi-regular (i.e. the majority of the vertices have valence 6).

2.2.3.1. New Triangle Creation

Given a candidate seeding position, to create a new triangle, we will need to create the base edge of the new triangle first. For extracted feature position, the adjacent feature information is always available and will be used to create the base edge. Figure 5 shows an illustration. For a boundary particle *m* with step size *s*, we will randomly create a line

segment *uv* of length $\frac{2\sqrt{3}}{3}s$ on the tangent plane of *m*, centered at *m* with two endpoints *u* and *v*, and then project the two points *u* and *v* onto the iso-surface as *u′* and *v′*, respectively. The new line segment *u′v′* will then serve as the base edge for this new triangle. Given a base edge *u′v′* and step size *s*, either obtained from the above seeding positions or from the boundary edge of an evolving front, to create a new triangle, we will need to create the third vertex. First the middle point *m* between *u′* and *v′* is calculated. A vertex *w* is then created at distance *s* away from *m* along the direction of the cross product between *u′v′* and $\overrightarrow{n_m}$. $\overrightarrow{n_m}$ is the outward-orienting unit vector obtained by the normalized gradient of the iso-surface at position *m*. *w* is then projected to *w′* on the iso-surface and will be the third vertex of the new triangle *u′v′w′*. Since the edge length of *u′v′* is $\frac{2\sqrt{3}}{3}s$, and the edge length of *mw* is *s*, hence the triangle *u′v′w* will be equilateral and the corresponding new triangle *u′v′w′* will be very close to equilateral.

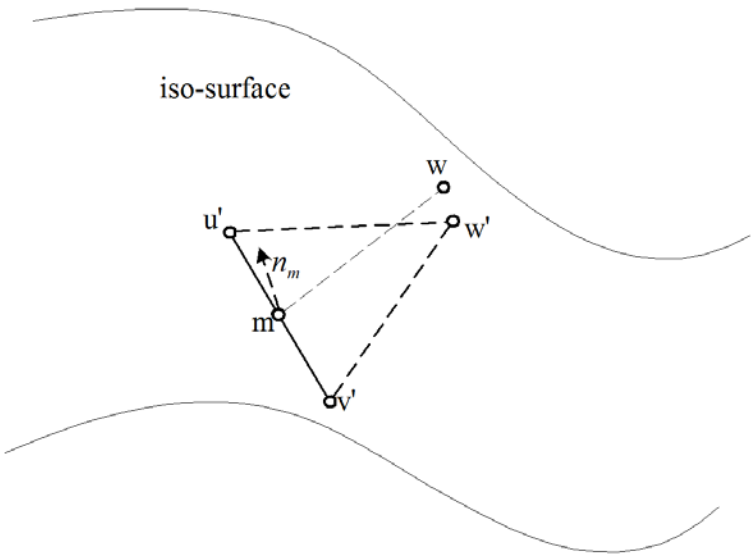


Figure 5 : New triangle creation

A new triangle *u'v'w'* is created by placing the third vertex *w* on the iso-surface (Section 2.2.3.1)

2.2.3.2. Projection

To project a given point onto the iso-surface (e.g. *w* and *w'* in Figure 5), we employed the iterative Newton-Raphson root-finding method as also used by Co et al. in [hama03]. Let $\phi(x)$ represents the density function of the volume data, the iso-surface of the volume data is then implicitly defined as $\{x \mid \phi(x) = 0\}$. The projection of a vertex *v* onto the iso-surface can be found by a few steps of iterations along the gradient direction $\nabla\phi(x)$ of the density function $\phi(x)$ until converge:

$$v_k = v_{k-1} - \frac{\phi(v_{k-1})}{\left\|\nabla\phi(v_{k-1})\right\|^2}\nabla\phi(v_{k-1}), k = 1,...n \quad (2.1)$$

with $v_0 = v$. The equilibrium position v_n obtained from this Newton-Raphson root finding method however, does not always stay on the iso-surface. Hence, we propose to conduct a finite number (e.g. three) iterations of additional line search after the Newton-Raphson root finding method converges to position v_n. More specifically, we will start from v_n and search along the gradient direction of v_n until two consecutive points on the searching sequence v_{n_i} and $v_{n_{i+1}}$ are found to be on the different sides of the iso-surface. A binary search is then conducted between the two points v_{n_i} and $v_{n_{i+1}}$ until the final projection point v_p is found, which can be as close to the iso-surface as needed. See Figure 6 for an illustration. If after a finite number of line search we can not find two consecutive points on the search direction that are on the opposite side of the iso-surface, the projection will be marked as invalid and the base edge will be freezed, which will be reactivated later at a smaller step size.

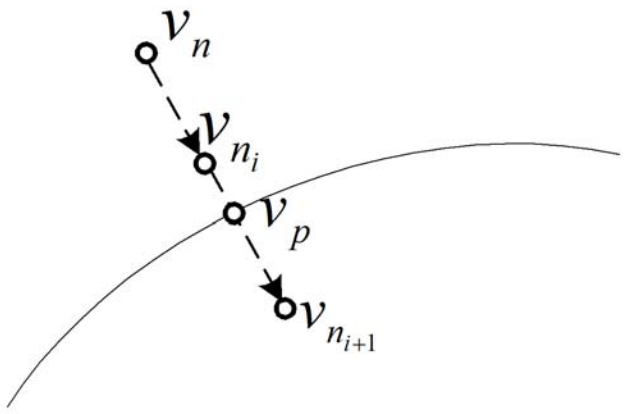


Figure 6 : Enhanced projection scheme

After the equilibrium position v_p is obtained from the Newton-Raphson root finding, several additional iterations of line search is conducted along the gradient of v_n to find the final projection point v_p which can be as close to the iso-surface as possible (Section 2.2.3.2).

2.2.4. Front Propagation

After a new triangle is created (Section 2.2.3.1), several tests need to be conducted before it can be added into the existing propagating front. One of the tests is to ensure the Delaunay face property, which is first proposed in the original Marching Triangles algorithm [hilt96]. However, the Delaunay face property will work only if the intersection of the Delaunay sphere and the iso-surface is a topological disk. It's an important property lacking in the original Marching Triangles algorithm [hilt96] and its existing variants [akko01, kark01, rodr04]. This problem is fixed by our new Normal Consistency Constraint to be explained later in this section.

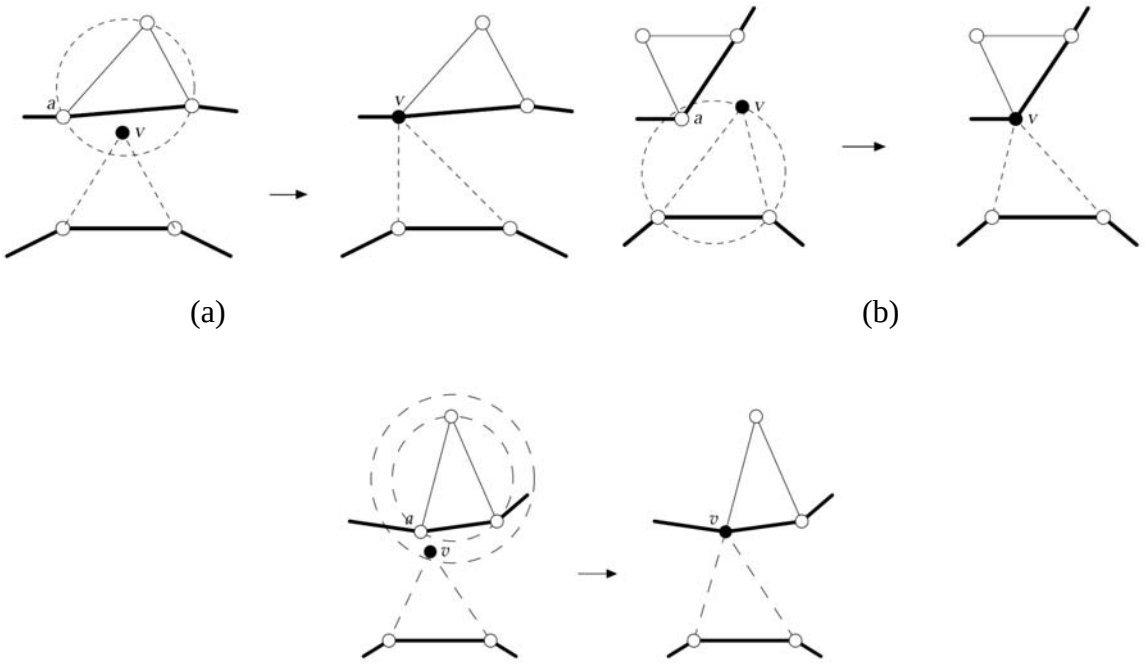


Figure 7 : Enforcing Delaunay face property

(a) New vertex *v* is inside the Delaunay sphere of an existing boundary triangle, and is snapped into the existing boundary vertex *a*; (b) The Delaunay sphere of the new triangle encloses an existing boundary vertex *a*, hence the vertex *v* is snapped into vertex *a*; (c) The new vertex *v* is very close to an existing boundary vertex *a*, hence it is snapped into vertex *a* to avoid creating a thin triangle passing through vertices *a* and *v*.

2.2.4.1. Delaunay Face Property

We will briefly review the Delaunay face property in this section. For more details, please refer to the original paper [hilt96]. In a nutshell, the Delaunay face property is: "A new triangle *xyv* with new vertex *v* can be created if and only if the Delaunay sphere of the triangle *xyv* does not contain any existing mesh vertices, and any Delaunay spheres of existing triangles do not contain the new vertex *v*". If all the triangles satisfy the

Delaunay face property, then the final triangulation will be the Delaunay triangulation, which is the optimal triangulation [hilt96].

To enforce the Delaunay face property, our algorithm performs the following two steps as illustrated in Figure 7. First, for new vertex *v*, we will check whether it is enclosed by the Delaunay spheres of any existing triangles. If yes, then one of such triangle's boundary vertices (e.g. *a*) will be used as the candidate for the third vertex of new triangle (Figure 7 (a)).

The second step (Figure 7 (b)) is to check whether there are any existing boundary vertices enclosed by the Delaunay sphere of the new triangle *xyv*. if yes, then one of such boundary vertices (e.g. *a*) will be used as the candidate and conduct this second step again until the Delaunay sphere of the new triangle does not include any existing vertices.

The above two tests however does not guarantee no thin triangles will be created as is illustrated by Figure 7 (c). Here the new vertex *v* is not included by any Delaunay sphere of existing triangles and the Delaunay sphere of the new triangle does not include any existing vertices. However since *v* is very close to *a*, a thin triangle (passing through *v* and *a*) will be created in later steps. To avoid creating such thin triangles and thus further improving the triangulation quality, in the first step, when checking whether the new vertex *v* is enclosed by the Delaunay spheres of any existing triangles, we will increase the radius of the sphere by an extra *d* (Figure 7 (c)). *d* is the user-defined minimum edge length of the triangles. By enlarging the sphere by an extra *d*, we ensure no thin triangles whose minimum edge length is less than *d* will be created. If the new triangle *xyv* passes these tests, it will be tested for the Normal Consistency Constraint to be explained in the following section.

2.2.4.2. Normal Consistency Constraint

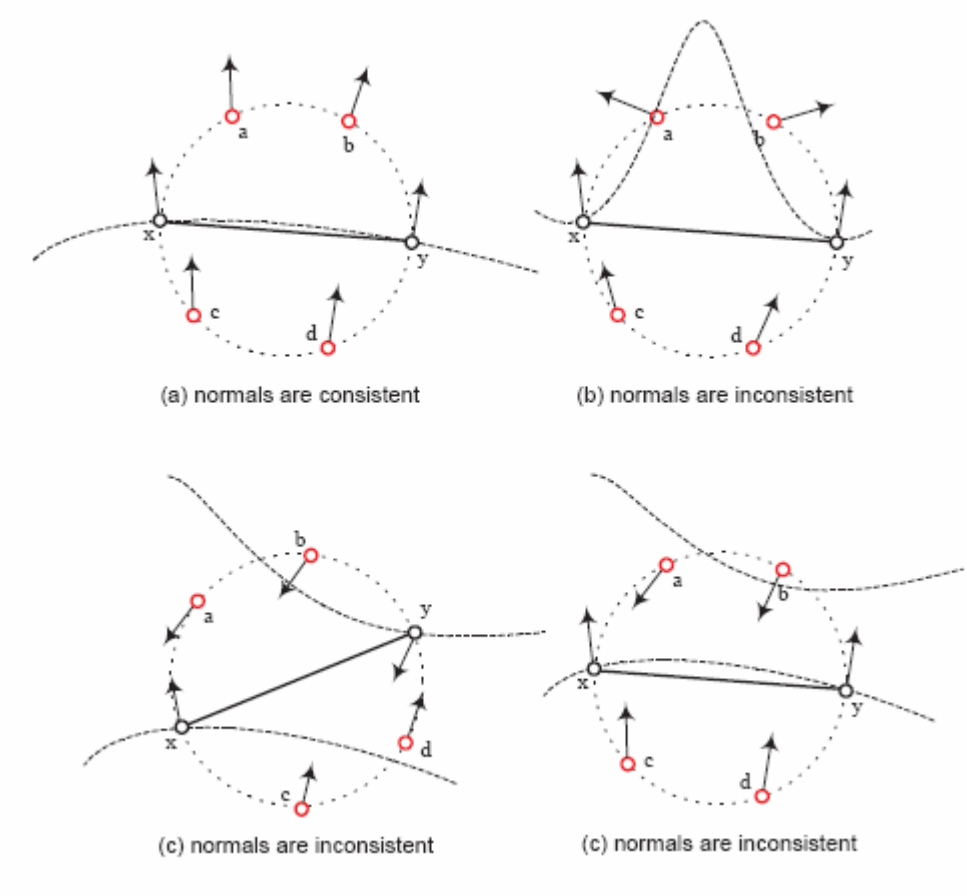


Figure 8 : A 2D illustration of the Normal Consistency Constraint

A new triangle (a new edge in this case) is valid only if it passes the Normal consistency constraint. This is used to ensure that the intersection of the Delaunay sphere (dashed circle) of the current edge (solid line) and the iso-surface (dashed curve) is a topological disk (Section 2.2.4.2). Among the four cases shown ((a)-(d)), only the edge in (a) passes the Normal consistent constraint.

The above Delaunay face property however will only work if the intersection of the Delaunay sphere and the iso-surface is a topological disk. One of the main contribution of this algorithm and the main difference between this algorithm and other Marching Triangles algorithms [hilt96, akko01, kark01, rodr04], is that we propose a specific

normal consistency checking step to enforce that the intersection of the Delaunay sphere of the new triangle and the iso-surface is a topological disk. More specifically, given a new triangle, we will uniformly sample its Delaunay sphere and calculate the maximum normal variation among the sampled positions. In particular, besides the three vertices of the new triangle, we will sample three extra points on each half of the Delaunay sphere divided by the new triangle. If the maximum normal variation among the sampled positions is less then a threshold, the new triangle is valid and can be added to existing mesh, otherwise the new triangle will not be created, and the base edge of the new triangle will be marked as inactive and will be freezed. Based on our experiment, the threshold of $\pi/3$ has worked very well. When all the boundary edges of a propagating front becomes inactive, the whole front will become inactive and will be inserted into the seeding priority list at a row one level lower than its current priority (Section 2.2.3). Figure 8 shows a 2D illustration of the Normal consistency constraint. Among the four cases shown (Figure 8 (a)- Figure 8 (d)), only the case in Figure 8 (a) passes the Normal Consistent Constraint, i.e. the intersection of the Delaunay sphere (dashed circle) of the current edge (solid line) and the iso-surface (dashed curve) is a topological disk.

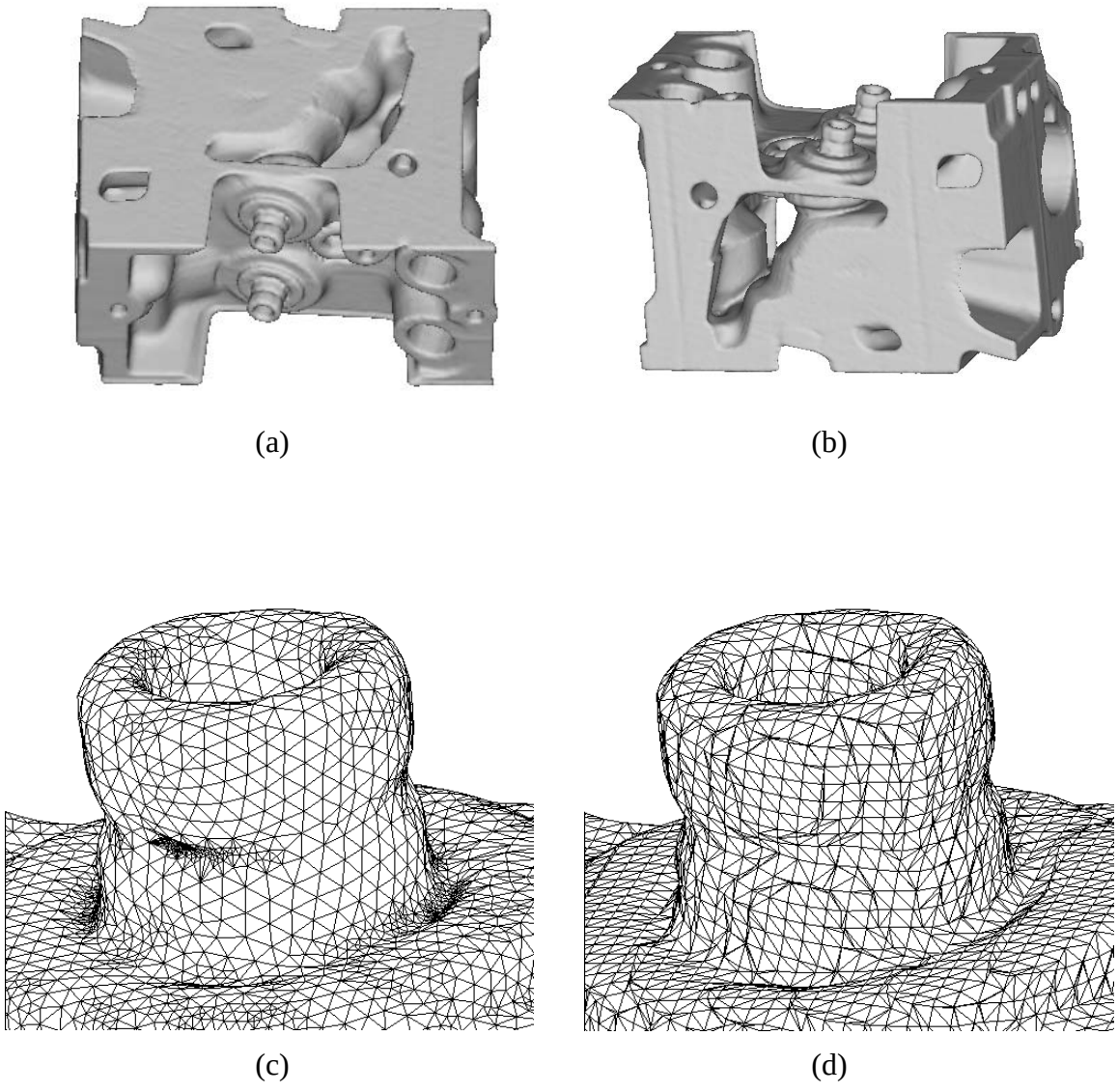


Figure 9 : Iso-surface extraction of a CT data of Engine

(a) and (b) are the front and back view of the smooth rendered mesh. (c) is a close-up view of mesh generated by our algorithm. (d) is a close-up view of mesh generated by Marching Cubes.

2.2.5. Identify Visited Boundary Particles

After a new triangle is created, all the boundary particles enclosed by the triangle's Delaunay sphere will be in the topological disk generated by the intersection of the Delaunay sphere and the iso-surface. These boundary particles will be marked as visited and will be removed from the seeding priority list (Section 2.2.3).

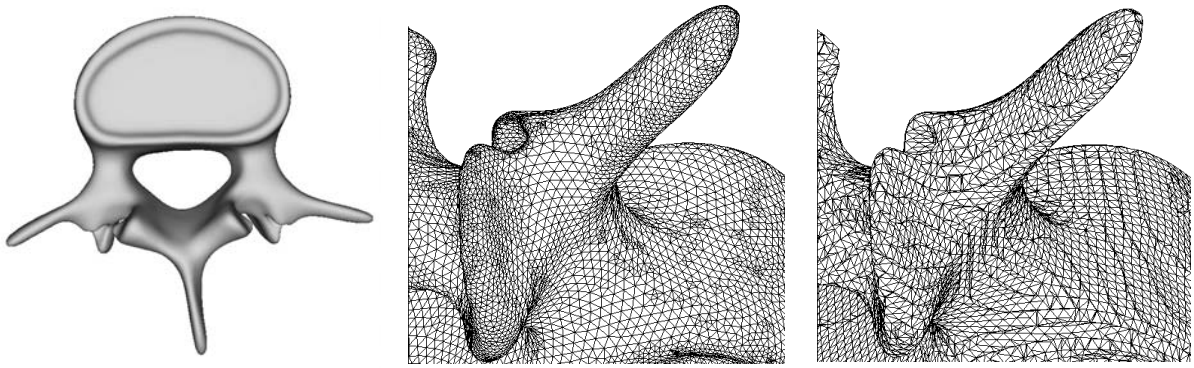
2.3. Experimental Results

Dataset	Volume Size	Times (second)	Vertices	Edges	Faces
dragon	149x129x89	95	26724	159807	53269
vertebra	132x124x56	218	62494	370938	123646
engine	256x256x128	1055	182598	1095522	365174
lobster	128x128x128	2520	549118	2803527	934509
brain	256x256x124	2835	638081	3613359	1204453

Table 1 : Running time information

Figure 1 and Figure 9 to Figure Figure 14 demonstrate some of the experimental results obtained by our new algorithm. The dark colored shape is rendered by smooth shading, while the green lines show the wire frame view of the mesh. For example, from Figure 10 (b), we can clearly see the high quality of the mesh that is adaptive to the local shape curvature. The new algorithm can also recover sharp features as can be seen in Figure 11. The sharp features are detected similar to [kobbelt01] and [ju02]. Figure 14 shows a side-by-side comparison between our new algorithm and the original marching triangle algorithm [hilt96]. As we can see, by enforcing normal consistency constraints proposed, our algorithm accurately extract the iso-surface in the regions around the small extrusion

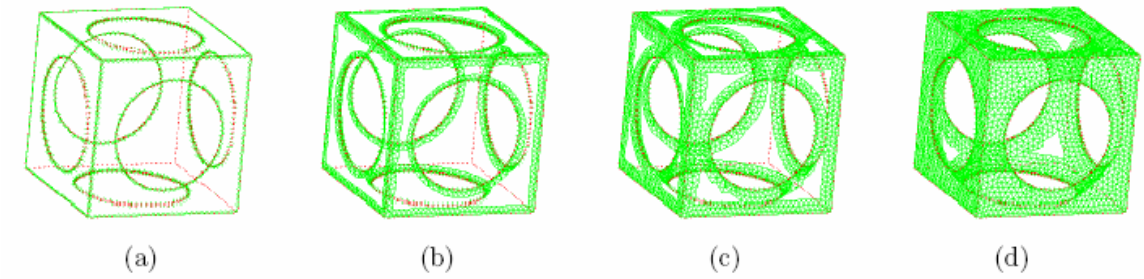
in the data (Figure 14 (a)). On the other hand, the original algorithm [hilt96] can not correctly extract the iso-surface and an self-intersection will be created (Figure 14 (b)). Table 1 summarizes the statistics of the examples, including the (x, y, z) dimensions of the input volume data, the model complexity (number of vertices, number of edges, and number of faces), and the running time (measured in seconds). All the experiments are conducted on a Pentium M 1.6GHZ Notebook PC with 1GB RAM. As can be seen from the running time table, the computational complexity of the new algorithm is linear of the size of the extracted mesh, which is similar to other region growing based algorithms [hilt96, akko01, kark01, rodr04]



(a) (b) (c)

Figure 10 : Iso-surface extraction of a CT data of vertebra phantom

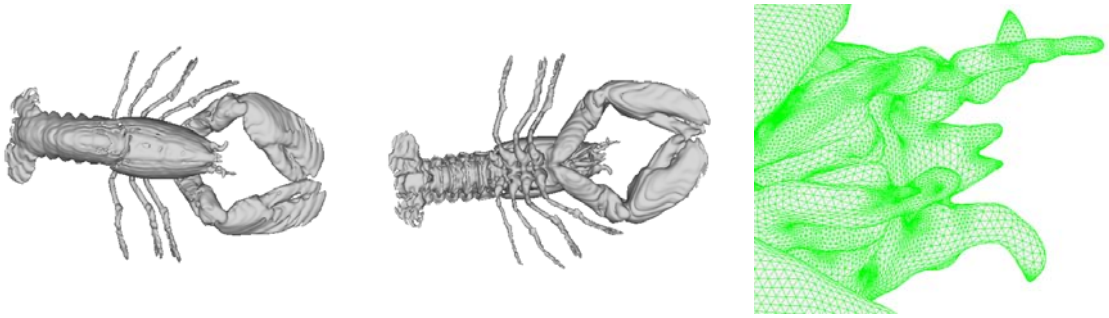
(a) smooth rendering of the extracted vertebra; (b) a close-up view; (c) a close-up view of mesh generated by Marching Cubes.



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

Figure 11 : Iso-surface extraction of a volume data with sharp edges (shown as red lines)

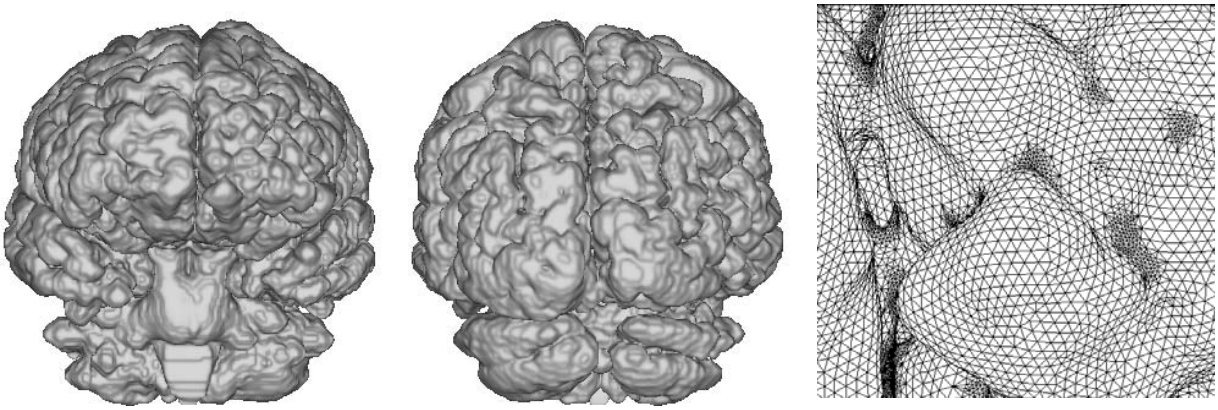
(a) is the seed initialization step. (b) and (c) are two snap shots during the front propagation. (d) is the final mesh.



(a) (b) (c)

Figure 12 : Iso-surface extraction of a CT data of a lobster

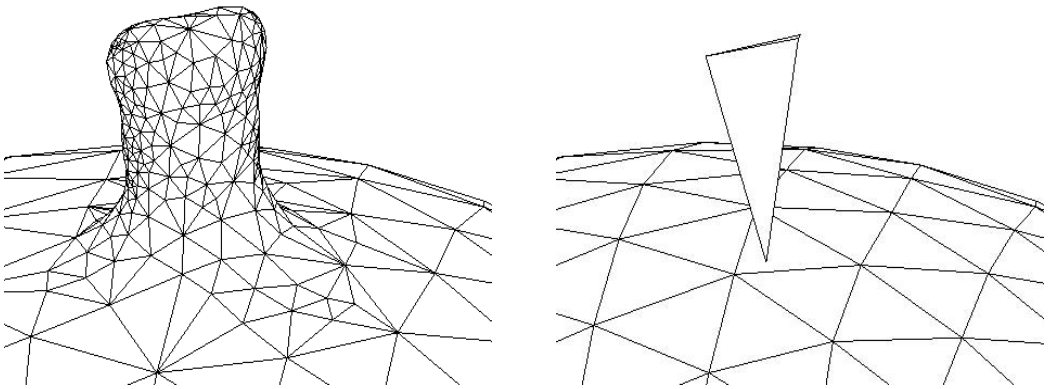
(a) and (b) are the front and back view of the smooth rendered mesh. (c) is a close-up view of (b).



(a) (b) (c)

Figure 13 : Iso-surface extraction of a MRI data of brain

(a) front view; (b) back view; (c) close up view of the generated mesh.



(a) (b)

Figure 14 : A comparison to the original marching triangle algorithm [hilt96].

By enforcing normal consistency constraints, our algorithm can accurately extract the iso-surface in the regions around the small extrusion in the data ((a)). On the other hand, the original algorithm [hilt96] can not correctly extract the iso-surface and an self-intersection will be created ((b)).

3 Nonparametric Noisy Points Preprocessing

3.1. Introduction

Despite significant advancement in interactive shape modeling, creating realistic looking 3D models from scratch is still a very challenging task. Recent advancement in 3D shape acquisition systems such as laser range scanner, structural light system and stereo vision system have made direct 3D data acquisition feasible [seit06].

The obtained point dataset however can be quite noisy. Without preprocessing to deal with noisy data, parametric and/or variational formulations [hopp92, baja95, curl96, hilt98, whit97, zhao00, zhao01, carr01, ohta03, xie03, levi03, alex03] are usually used. These methods generally composed of both a fitting term for the data and a regularization term for the reconstructed surface. Since all data points, even outliers, are treated equally and can affect the final reconstruction, these approaches will likely fail for highly noisy data.

Another type of algorithms is based on popular computational geometry algorithms such as Delaunay triangulations and Voronoi diagrams to construct triangulated surfaces [amen99, edel98, bois84, amen98, amen98a]. For this kind of method, it is challenging to find the right connections among all data points in three and higher dimensions, especially for noisy data.

Recently, Medioni *et al.* proposed the tensor voting method [medi00], which is a nice feature extraction algorithm. By designing an appropriate voting procedure among

all data points, a tensor field and an associated saliency field can be constructed. Coherent geometric information can be extracted from the tensor field and the saliency field. However, tensor voting method is computationally very expensive.

We propose a nonparametric approach for noisy point data preprocessing. In particular, we proposed an anisotropic kernel based nonparametric density estimation method for outlier removal, and a hill-climbing line search approach for projecting data points onto the real surface boundary. Our approach is simple, robust and efficient. We demonstrate our method on both real and synthetic point datasets.

3.2. Algorithm

3.2.1. Parzen-window Based Kernel Density Estimation

Points outside the object surface are outliers that have to be removed. Since the real object surface is unknown, it is hard to specify a general criterion to detect outliers. We propose to employ parzen-window based nonparametric density estimation method for outlier removal.

Parzen window based kernel density estimation is the most popular nonparametric density estimation method. In order to make the thesis self-contained, we will briefly review the Parzen window method in the following. For a complete description, please refer to the book by Duda *et al.* [duda00].

Given n data points x_i, $i = 1, \ldots, n$ in the d-dimensional Euclidean space R^d, the multivariate kernel density estimate obtained with kernel $K(x)$ and window radius h, computed in the point x is defined as:

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \qquad 3.1$$

Without loss of generality, let's assume h = 1 from now on, so we could simplify Equation (3.1) as:

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} K(x - x_i) \qquad 3.2$$

$K(x)$ is generally a spherically symmetric kernel function satisfying:

$$K(x) = C_{k,d} k(\| x \|^2) \qquad 3.3$$

where $k(x)$ is the profile function of the kernel $K(x)$. $C_{k,d}$ is the normalizing constant such that

$$K(x) \geq 0 \qquad 3.4$$

and

$$\int_{R^d} K(x) dx = 1 \qquad 3.5$$

$||x||$ is the L_2 norm (i.e. Euclidean distance metric) of the d-dimensional vector x. Employing the profile function notation, Equation (3.2) can be further rewritten as

$$f(x) = \frac{C_{k,d}}{n} \sum_{i=1}^{n} k(\| x - x_i \|^2) \qquad 3.6$$

There are three types of commonly used spherical kernel functions $K(x)$: the Epanechnikov kernel, the uniform kernel, and the Gaussian kernel. The Epanechnikov kernel is defined by the profile function $k_E(x)$:

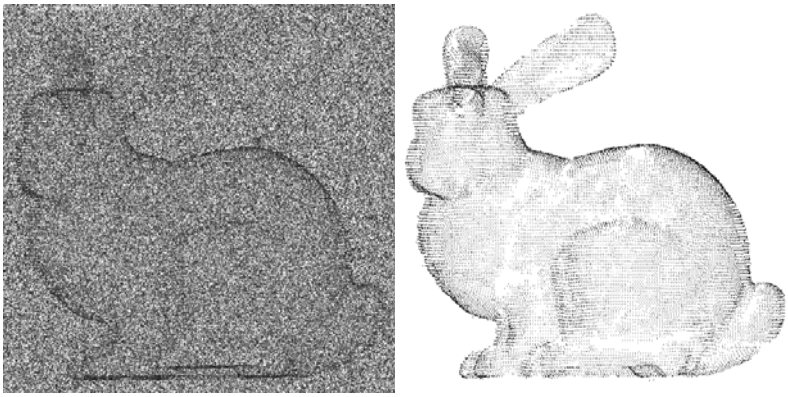
$$k_E(x) = \begin{cases} 1 - x & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases} \qquad 3.7$$

The uniform kernel is defined by the profile function $k_U(x)$:

$$k_E(x) = \begin{cases} 1 & 0 \le x \le 1 \\ 0 & x > 1 \end{cases} \qquad 3.8$$

The Gaussian kernel is defined by the profile function $k_N(x)$:

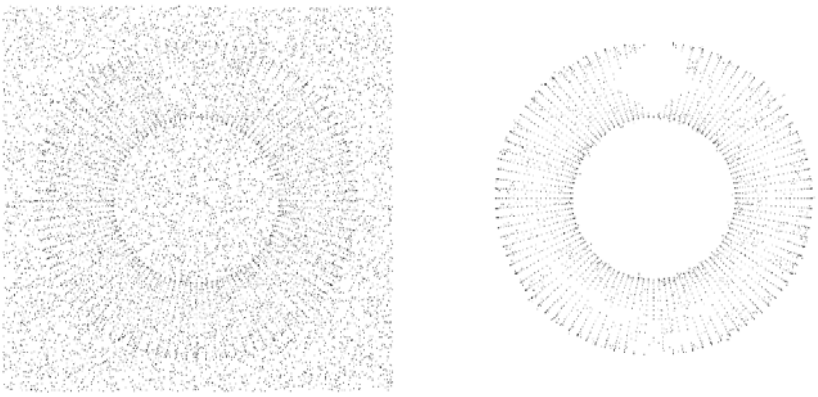
$$k_N(x) = \exp(-\frac{1}{2}x) \qquad x \ge 0 \qquad 3.9$$



(a) (b)

Figure 15 : Stanford bunny

The Stanford bunny (35678 points) with 1000% noise points added in; (b) Result after outlier removal.



(a) (b)

Figure 16 : Synthetic torus

(a) A synthetic torus point data (4800 points) which has an open hole in top, with 1000% noise points added in; (b) Result after outlier removal.

3.2.2. Outlier Removal By Anisotropic Ellipsoidal Kernel

For 3D point cloud obtained by laser scan or stereo vision, the outliers tend to spread in the space randomly, while “real” (we use a quotation here to emphasize the fact that the real surface is unknown) surface points will spread along a thin shell which encloses the real surface object. In other words, the distribution of the outliers is relatively isotropic, while the distribution of the real surface points is rather anisotropic. Hence, we propose to employ an anisotropic ellipsoidal kernel based density estimation method for outlier removal.

For anisotropic kernel, the L_2 norm $||x\text{-}x_i||$ in Equation (3.6), which measures the Euclidean distance metric between two points x and x_i will be replaced by the Mahalanobis distance metric $||x\text{-}x_i||_M$:

$$|| x - x_i ||_M = ((x - x_i)^t H^{-1} (x - x_i))^{1/2} \qquad 3.10$$

here H is the covariance matrix defined as:

$$H = DD^T \qquad 3.11$$

and

$$D = (x_1 - x, x_2 - x, \ldots, x_n - x) \qquad 3.12$$

Geometrically, $(x - x_i)^t H^{-1}(x - x_i) = 1$ is a three-dimensional ellipsoid centered at x, with its shape and orientation defined by H. Using Single Value Decomposition (SVD), the covariance matrix H can be further decomposed as:

$$H = UAU^T \qquad 3.13$$

with

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \qquad 3.14$$

$\lambda_1 \geq \lambda_2 \geq \lambda_3$ are the three eigenvalues of the matrix H, and U is an orthonormal matrix whose columns are the eigenvectors of matrix H. We will detail the SVD-based decomposition (Equation (3.13)) of the covariance matrix H in section 0.

To compute the anisotropic kernel based density, we will apply an ellipsoidal kernel E of equal size and shape on all the data points. The orientation of the ellipsoidal kernel E will be determined locally. More specifically, given a point x, we will calculate its covariance matrix H by points located in its local spherical neighborhood of a fixed radius. Without loss of generality, we will assume the radius is 1 (which can be done by normalizing the data by the radius). The U matrix of Equation (3.13) calculated by the covariance analysis is kept unchanged to maintain the orientation of the ellipsoid. The

size and shape of the ellipsoid will be modified to be the same as the ellipsoidal kernel E by modifying the diagonal matrix A as:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & r \end{bmatrix} \qquad 3.15$$

r is half of the length of the minimum axis of the ellipsoidal kernel E. After the density value is estimated, we will remove all the points whose estimated density value is smaller than a user-defined threshold. Figure 15 and Figure 16 are two examples of the proposed nonparametric density estimation based outlier removal from synthetic highly noisy data. The input of Figure 19 is a 3D point clouds obtained from multi-view stereo. Figure 19 (a) is the original data; Figure 19 (b) is the result after outlier removal. Figure 17 is a 2D slice view of the nonparametric data preprocessing process. In particular, Figure 17 (a) is a 2D slice of the original 3D data of Figure 19 (a); Figure 17 (b) is the color-coded density map of Figure 17 (a) estimated by an isotropic Gaussian kernel, with red represents the highest density and blue the lowest density; Figure 17 (c) is the color-coded density map of Figure 17 (a) estimated by an anisotropic Gaussian kernel of the same scale of Figure 17 (b); Figure 17 (d) is the result after outlier removal using Figure 17 (b), and Figure 17 (e) is the result after outlier removal using Figure 17 (c). As we can see, better result is achieved by using the anisotropic kernel-based outlier removal (Figure 17 (e)), comparing with the isotropic kernel-based outlier removal (Figure 17 (d)).

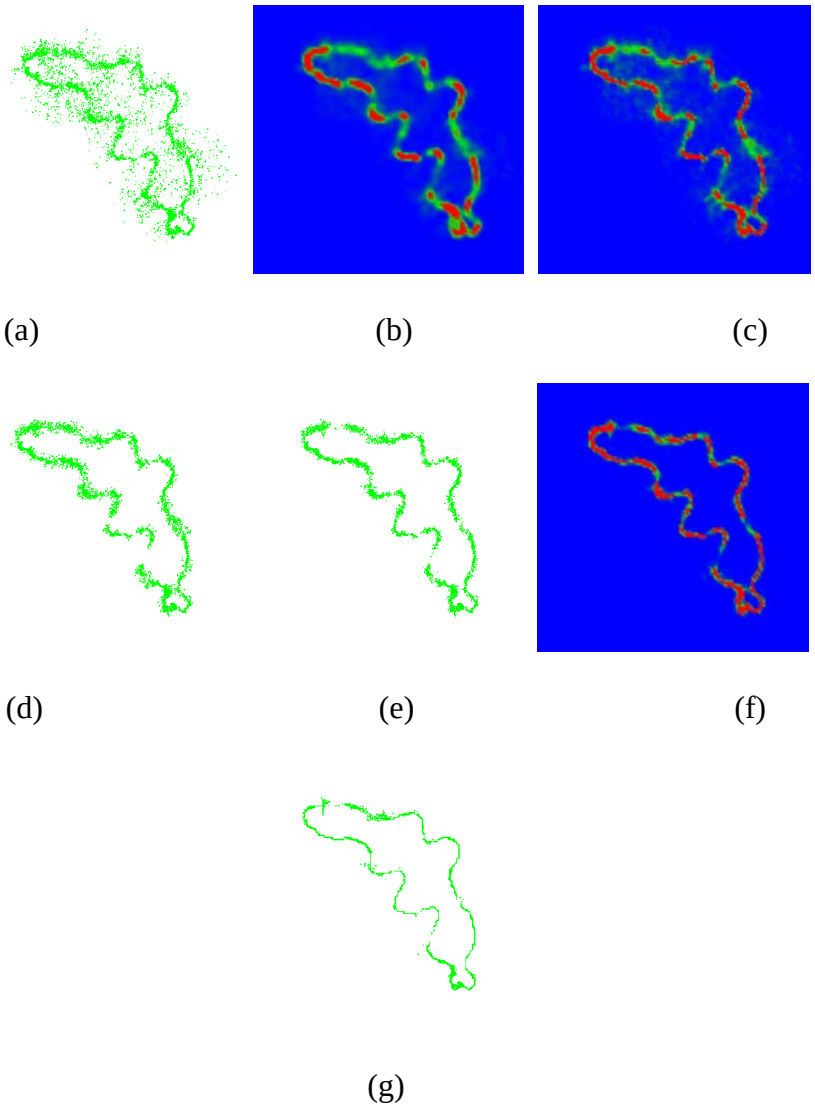


Figure 17 : A 2D slice view of the nonparametric data preprocessing process

(a) a 2D slice of the original 3D data of Figure 19 (a); (b) color-coded density map of (a) estimated by an isotropic Gaussian kernel, with red represents the highest density and blue the lowest density; (c) color-coded density map of (a) estimated by an anisotropic Gaussian kernel of the same scale of (b); (d) after outlier removal using (b); (e) after outlier removal using (c); (f) color-coded density map of (e) estimated by an anisotropic Gaussian kernel of a smaller scale than (c); (g) result of projecting points of (e) by line search based on (f).

Figure 18 shows the 3D outlier removal results under different user-defined thresholds. The top left is the original point clouds obtained by the aforementioned depth estimation step. The next four images from top middle to bottom right are the outlier removal results under different user-defined thresholds: 40, 60, 80, and 160, respectively. Among these four outlier removal results, the first three data (top middle to bottom left) are all acceptable to the subsequent implicit surface evolution step (Section 4.2.4) to construct a watertight 3D surface. However the implicit surface evolution step might fail to create a single watertight surface of the object for the fourth data in the bottom right of the figure as the threshold is set too high thus creating very big holes in the data.

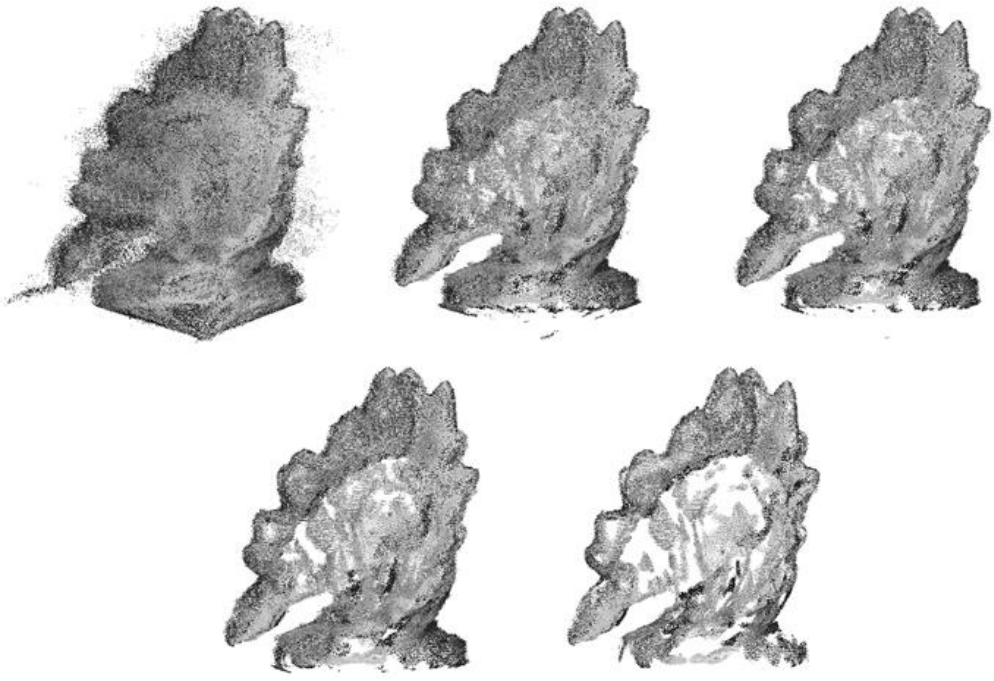


Figure 18 : Outlier removal results under different user-defined thresholds

Top left: original points obtained by depth estimation from the dino sparse ring data. From top middle to bottom right are the outlier removal results under different user-defined thresholds: 40, 60, 80, and 160, respectively.

3.2.2.1. Decomposition of the Covariance Matrix H

Since the covariance matrix H is defined as:

$$H = DD^T,$$

and

$$D = (x_1 - x, x_2 - x, \ldots, x_n - x).$$

Using Single Value Decomposition (SVD), D can be decomposed as:

$$D = U\Sigma V^T \qquad 3.16$$

Σ is the diagonal matrix:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \qquad 3.17$$

with $\sigma_1 \geq \sigma_2 \geq \sigma_3$. U and V are orthonormal matrixes, i.e. $U^T U = V^T V = I$, I is the identity matrix. So,

$$H = DD^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T \qquad 3.18$$

and let's define matrix A as:

$$A = \Sigma^2 \qquad 3.19$$

which is also a diagonal matrix. Thus we have

$$H = UAU^T \qquad 3.20$$

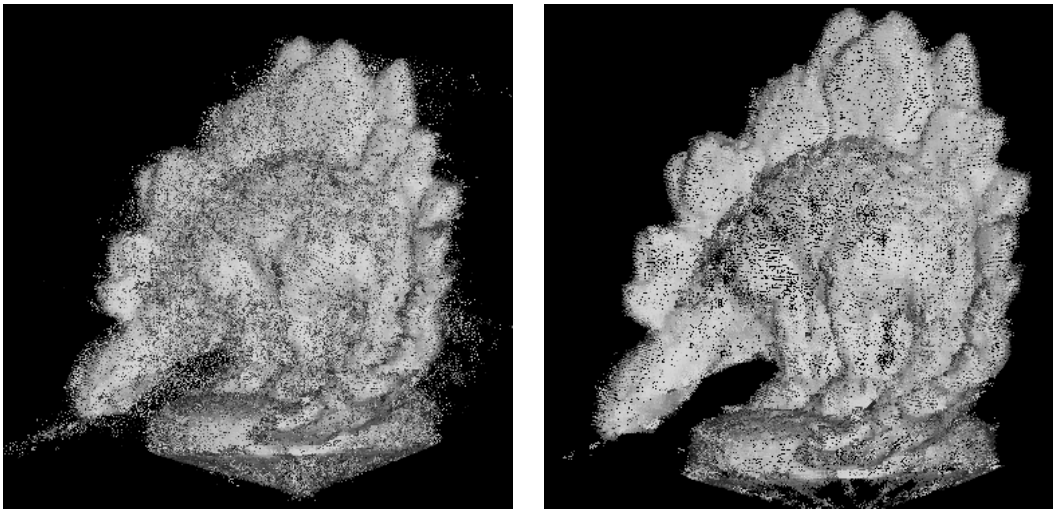
Furthermore, if we denote $U = (u_1, u_2, u_3)$, since U is an orthonormal matrix, we will have

$$Hu_i = \sigma_i^2 u_i \qquad 3.21$$

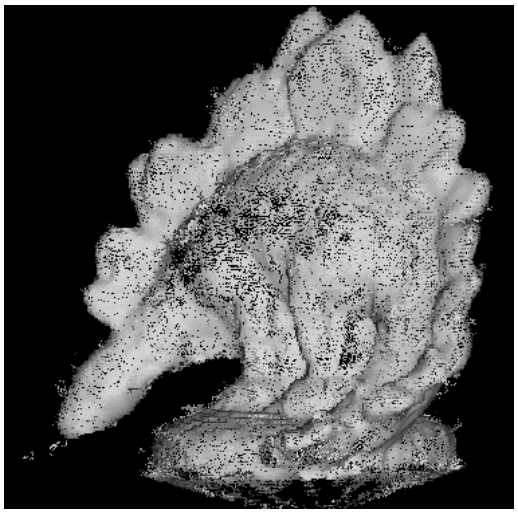
which means u_i is the eigenvector of H, associated with its i^{th} largest eigenvalue σ_i^2.

3.2.3. Point Projection By Line Search

After the outlier removal step, we will need to further process the remaining point data so that they will be projected into the real surface. We will define the real surface as the set of points whose density value (estimated by the above Parzen-window based kernel density estimation) is maximum along its surface normal direction. This is similar to the concept of "extreme surface" as suggested by Amenta *et al.* [amen04]. We implement the data projection by a hill-climbing line search method. More specifically, given a point *x*, it will move along a direction *l* that is (approximately) orthogonal to the real surface, and will stop when a local maximum of the density value is reached. Since the real surface is unknown, the moving direction *l* is approximated by the eigenvector u_3 (Equation (3.21)) that is associated with the smallest eigenvalue σ_3^2 (Equation (3.21)) of the covariance matrix *H* (Equation (3.11)), calculated by the local neighborhood of point *x*. The orientation of the moving direction is determined by the directional derivative of the density value along *l*, i.e. point *x* will always move uphill (i.e. move towards the direction that will increase the density value). Figure 19(c) shows the result of projecting data points by line search. A 2D slice view of the point projection is shown in Figure 17(g). Note that the outlier removal and the point projection procedure can be iterated one or two times to further improve the result.



(a) (b)



(c)

Figure 19 : Dino

(a) 3D point clouds (660,000 points) obtained from multi-view stereo; (b) after outlier removal; (c) after point projection by line search.

4 Multi-view Stereo

4.1. Introduction

Despite significant advancement in interactive shape modeling, creating complex high quality realistic looking 3D models from scratch is still a very challenging task. Recent advancement in 3D shape acquisition systems such as laser range scanners and encoded light projecting system have made directly 3D data acquisition feasible [wang04]. These active 3D acquisition systems however remain expensive. Meanwhile, the price of digital cameras and digital video cameras keeps decreasing while the quality is improving every day, partially due to the intense competition in the huge consumer market. Furthermore, huge amounts of images and videos are added in internet sites such as Google ®, etc every day, a lot of which could be used for multiview image-based 3D shape reconstruction [goes07].

To date, there have been a lot of researches conducted in the area of multiview image-based modeling. The recent survey by Seitz et al. [seit06] gives an excellent review of the state-of-arts in this area. As summarized by [camp08], most of the existing algorithms follow a two-stage approach: 1) conduct depth estimation based on local groups of input images; 2) fuse the estimated depth values into a global watertight 3D surface estimation. The depth estimation step is often based on image correlation [hern04]. The main differences between existing algorithms are in the second stage, the data fusion step, which can be divided into two categories. The first type of data fusion

reconstructs the 3D surface by conducting volumetric data segmentation using global energy minimization approaches such as graph cut [vogi07, goes06, horn06, vogi05, sinh05], level-set [jim05, faug98, soat03, jin03, maxi05, kolm02], or deformable models [hern04, duan04, hern02, yasu06]. Recently, people have proposed another types of data fusion algorithms that are based on local surface growing and filtering [goes07, furu07, habb07]. Without global optimization, these types of data fusion algorithms can be computationally more efficient [merr07, brad08].

Our algorithm also follows this two-stage process. We proposed two iterative refinement schemes that iterates between the depth estimation step and the data fusion step. This is similar in spirit of the Expectation Maximization (EM) algorithm. The difference between these two algorithms is in implicit and explicit surface evolution. One uses outliner removal algorithm to clean points cloud, followed by tagging algorithm for implicit surface evolution, and energy-optimization based partial differential equations (PDE) for explicit surface evolution; another uses integrate graph-cut method for global optimal initialization with flux tensor based divergence field in implicit surface evolution, and mean shift for explicit surface evolution. Our algorithms integrate the fast implicit region growing with the high-quality explicit surface evolution, thus they are both fast and accurate.

The rest of this chapter is organized as follows: In Section 1.1 we discuss the main differences between our approaches and related existing works. Section 2 and 3 describe the details of our algorithms and show their benchmark data evaluations.

4.1.1. Comparison With Related Works

Our work is most related to the works of Hernandez et al. [hern04] and Quan et al. [quan07, maxi05]. Hernandez et al. proposed a deformable model based reconstruction algorithm [hern04] that achieves one of the highest quality reconstruction [seit06]. The depth estimation of [hern04] is conducted by rectangular window based normalized cross-correlation (NCC). The estimated depth values are then discretized into an octree-based volumetric grid. Finally a gradient vector flow based deformable model is applied to the volumetric grid to reconstruct the 3D surface.

Our depth estimation follows the similar pipeline of [hern04], with several modifications to further improve its efficiency and accuracy. We will describe these modifications in Section 2.2. Furthermore, unlike [hern04], we represent the depth estimations as 3D points whose accuracy are not restricted by the resolution of the volumetric grid. Quan et al [quan07, maxi05] also represent the estimated depth values as 3D points. However, unlike our method, they do not have an explicit outlier removal. Instead they rely on level-set based surface evolution with high-order smoothness terms such as Gaussian/mean curvature to overcome noises, which may create surfaces that maybe too smooth to represent finer geometry details of the original object. Most recently, Campbell et al. [camp08] proposed an outlier removal algorithm based on the Markov Random Field (MRF) model which can achieve very impressive reconstruction results. On the other hand, our outlier removal algorithm is based on kernel density estimation and is conducted on 3D unorganized points instead of the 2D image space of [camp08].

The graph cut algorithm as well as the divergence based energy functional are originally proposed by Lempitsky et al. [vict07]. However the way we created the

divergence field is totally different than [vict07]. Also [vict07] is focused on surface fitting with multi-view stereo one of the application, where we proposed a complete pipeline for multi-view stereo in this thesis.

To summarize, the main contributions of this thesis are: 1) a novel iterative refinement scheme between the depth estimation and the data fusion; 2) a novel anisotropic kernel density estimation based outlier removal algorithm and saliency estimation algorithm; 3) a novel data fusion approach that integrates the global optimal graph cut algorithm with a high-quality, efficient mean-shift based explicit surface evolution algorithm.

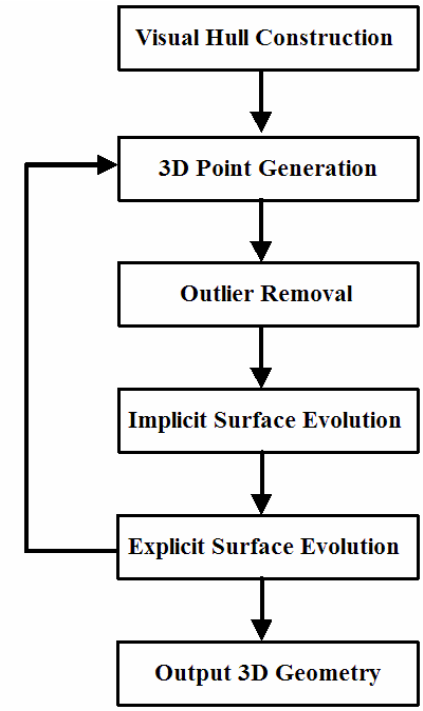


Figure 20 : Flow chart of the algorithm with outliner removal.

4.2. Iterative Surface Evolution Algorithm With Outliner Removal

The entire algorithm (Figure 1) consists of the following main steps:

1. Visual hull construction
2. 3D point generation
3. Outlier removal
4. Implicit surface evolution
5. Explicit surface evolution

Starting from an initial shape estimation such as the visual hull (Step 1), we will use this shape estimation to generate more accurate 3d points based on image correlation based depth estimation (Step 2), which can then be used to create a better shape estimation (Step 3 to Step 5). In practice, two to three iterations between Step 2 and Step 5 will be sufficient to create very good shape estimation. Figure 21 is a 2d illustration of the reconstruction process.

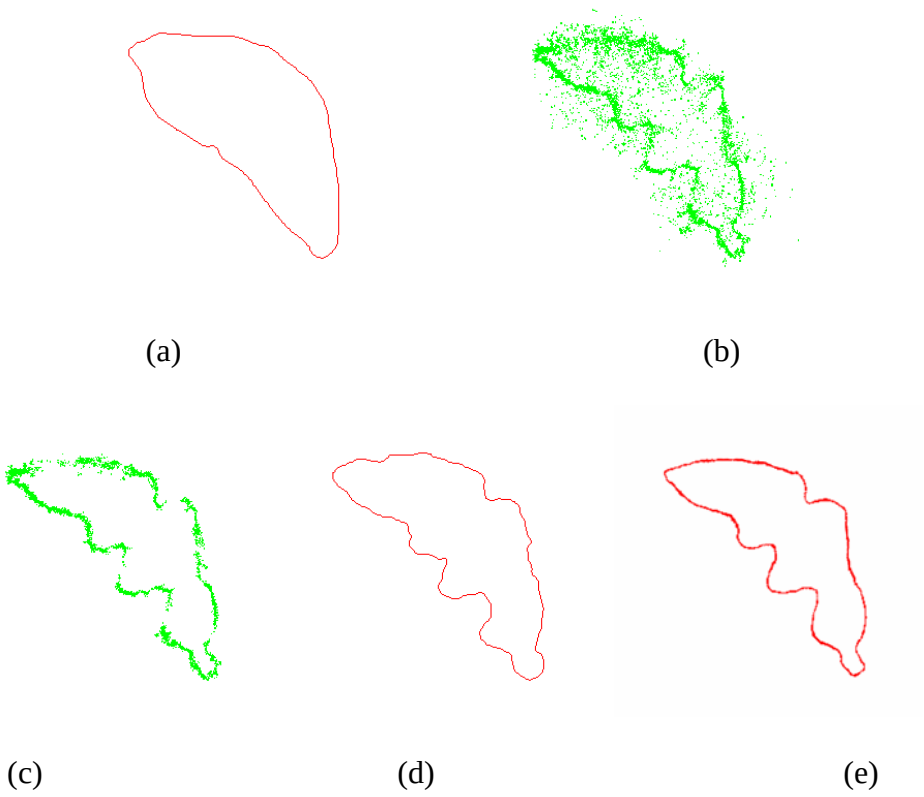


Figure 21 : A 2D illustration of the whole reconstruction pipeline

(a) visual hull; (b) 3d points generated by depth estimation; (c) after outlier removal; (d) shape estimated by implicit region growing; (e) refined shape estimation by explicit surface evolution.

4.2.1. Visual Hull Construction

The first step of our algorithm is to obtain initial shape estimation by constructing a visual hull. Visual hull is an outer approximation of the observed solid constructed as the intersection of the visual cones associated with all the input cameras [laur94]. A discrete volumetric representation of the visual hull can be obtained by intersecting the cones generated by back projecting the object silhouettes from different camera views. An explicit shape representation can be obtained by iso-surface extraction algorithms such as Marching Cubes [lorensen87].

4.2.2. 3D Points Generation

Once we had an initial explicit shape estimation, we will proceed to 3D depth estimation. First, we need to estimate the visibility of the initial shape with respect to all the cameras. We use OpenGL to render the explicit surface into the image planes of each individual camera and extract the depth values from the Z-buffer. Given a point on the surface, its visibility with respect to a given camera can then be decided by comparing its projected depth value into the image plane of the given camera with the corresponding depth value stored in the Z-buffer.

Our depth estimation is based on the Lambertian assumption, i.e. if a point belongs to the object surface, its corresponding 2D patches in the image planes of its visible cameras should be strongly correlated. Hence starting from a point on the object surface, we can conduct a line search along a defined search direction to locate the best position whose correlation between the corresponding 2D image patches of different visible cameras is the maxima within a certain search range. This idea is first proposed by [hern04]. Our thesis follows the same principle with several modifications. In the following, we will first briefly describe our depth estimation method, followed by a brief discussion of the differences between our method and the method of [hern04].

Given a point on the initial surface, we will select a set of (up to) five “best-view” visible cameras based on the point’s estimated surface normal. Each camera in the selected set will serve as the main camera for once. The search direction is defined as the optical ray passing through the optical center of the main camera and the given point. We will uniformly sample the optical ray within a certain range of the given point, and for each sampled position, we will project it into the image planes of the main camera and

another camera in the set, respectively. Rectangular image patches centered at the projected locations of the two image planes will be extracted, and the correlation between the two image patches will be computed by similarity measures such as the normalized cross-correlation (NCC) [hern04].

For a set of five “best-view” cameras, a total of 20 correlation curves will be generated. For each of the correlation curve, the best position (i.e. the point with the highest correlation value) will be selected as the depth estimation. The depth estimations will be represented as 3d points, which will be processed further to construct a new shape estimation of the object.

The main differences between our implementation and the method of [hern04] are: first, we start the line search from every point on the explicit object surface. The line search in [hern04] is initiated from every image and the correlation is computed with all the other images, which could be computationally more expensive then ours. Secondly, in [hern04], for each set of correlation curves computed using the same search direction and the same main camera, only one representative depth estimation is used. While in our method, we avoid this potentially premature averaging by using the depth estimations from all the correlation curves, and postpone the outlier pruning into the subsequent outlier removal step. Thirdly, in [hern04], the depth estimations are stored in an octree-based volumetric grid, while we store them as discrete points whose accuracy are not restricted by the grid size.

4.2.3. Outlier Removal

We use the algorithm detailed at section 3.2.1 and section 3.2.2 for outlier removel. Please refer to those sections for detail.

4.2.4. Implicit Surface Evolution

After outlier removal, the remaining 3d points will be used to reconstruct the 3D surface of the object. The shape estimation is conducted into two steps. First, a fast implicit distance function based region growing method—tagging algorithm [zhao01] is employed to create a coarse shape estimation from the 3d points. Next, an explicit surface evolution step is applied to recover the finer geometry details of the object. We will briefly review the tagging algorithm in the following, for more details please refer to the original paper [zhao01]. The explicit surface evolution method will be discussed in the next section.

The basic idea of tagging algorithm is to identify as many correct exterior grid points as possible and hence provide a good initial implicit surface, which is represented as an interface that separates the exterior grid points from the interior grid points. There are two main steps in the original tagging algorithm. First, we will compute a volumetric unsigned distance field based on the 3d points. This is done by the aforementioned fast sweeping method [zhao00]. Once we had the volumetric unsigned distance field, the tagging algorithm will iteratively grow the set of exterior grid points and stop at the boundary of the object. The algorithm can start from any initial exterior region that is a subset of the true exterior region, e.g. an outmost corner grid point of the bounding volume, and iteratively tag all the grid points as exterior or interior points based on the

comparison of the closeness to the object boundary between the current grid points and its neighboring interior grid points.

4.2.5. Explicit Surface Evolution

The shape estimation obtained by the implicit tagging algorithm will be converted to explicit meshes by [lorensen87], and will serve as the initial shape for the subsequent explicit surface evolution step to further improve the geometry accuracy of the shape reconstruction. The surface evolution is guided by energy-optimization based partial differential equations (PDE). Classical PDEs such as minimal surface flow [case96] usually includes a second order curvature term to improve the robustness against noise. However it may also prevent the surface evolution to recover finer geometry details. In this thesis, we choose the simple convection equation to guide the explicit surface evolution:

$$\frac{\partial S}{\partial t} = g(S)\vec{N}, \qquad g(S) = f'(S) \tag{4.1}$$

$S = S(t)$ is the 3D evolving surface, t is the time parameter, $g(S)$ is speed function and is defined as the derivative of $f(S)$, which is the point-based density estimation calculated by equation (3.1). $\vec{N}$ is the surface normal vector. The final reconstructed 3D shape is then given by the steady-state solution of the equation: $S_t = 0$.

Since the speed function g is dynamically calculated at each time step based on the local points distribution, the accuracy of our evolution method will not be limited by the grid resolution as other volumetric image based surface evolution methods such as [hern04]

4.2.6. Benchmark Data Evaluation

We had applied this algorithm to the four benchmark datasets: temple ring, temple sparse ring, dino ring, and dino sparse ring from [multiv]. Table 2 shows the running time and the reconstruction accuracy obtained from the evaluation site [multiv]. The running time is based on a Pentium D Desktop PC with CPU 2.66GHz, 2GB RAM. Figure 22 are the 3D rendering of our reconstruction results copied from the evaluation website. Note that, the evaluation was originally done in the private evaluation mode, thus we can't give out the link in this thesis. We are contacting Dr. Daniel Scharstein to move our results into the public domain.

Dataset	Running time (minutes: seconds)	# of input images	accuracy
Temple ring	33:17	47	98.9%
Temple Sparse Ring	29:06	16	96.8%
Dino ring	36:45	48	97.7%
Dino sparse ring	32:01	16	97.6%

Table 2 : Running time and reconstruction accuracy

Figure 22 : Temple and Dino 3D illustration

3D rendering of our reconstruction results running on the four Benchmark datasets of [multiv].

From top to bottom: temple ring, temple sparse ring, dino ring, and dino sparse ring.

4.3. Integrated Depth Fusion Algorithm with Graph-Cut

The entire algorithm consists of the following five main steps :

1. Visual hull construction
2. 3D point generation
3. Saliency Weighted Normal Vector Field Construction
4. 3D Shape Estimation by Graph-Cut
5. Explicit surface evolution by mean-shift

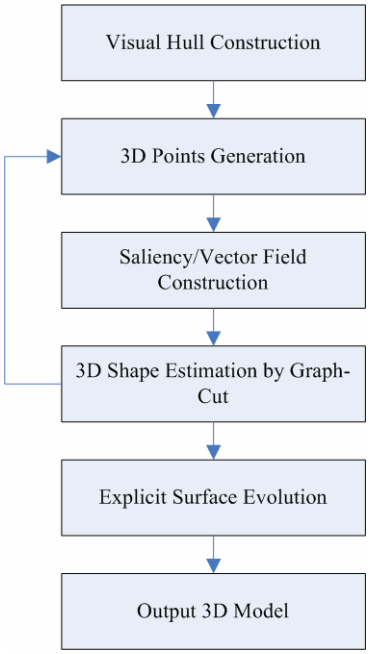


Figure 23 : Flow chart of the algorithm with Graph-Cutl

Starting from an initial shape estimation such as the visual hull (Step 1), we will conduct depth estimation which will generate a set of 3D points representing the estimated depth (Step 2). A volumetric saliency weighted normal vector field is then constructed based on the 3D points (Step 3), which is then be used to extract a watertight 3D surface by graph cut algorithm (Step 4). Step 2 to Step 4 can be repeated several times. In practice, two to three iterations between Step 2 and Step 4 will be sufficient to obtain a good shape estimation, which will be further improved by the explicit surface evolution step (Step 5). We will discuss each step in details in the following sections. Figure 24 is a 2D slice view of the reconstruction process of the dino ring dataset of [19]. Figure 25 to Figure 28 show the intermediate results of the 3D reconstruction process of dino sparse ring, dino ring, temple sparse ring, temple ring.

Instead of using outliner removal and tagging algorithm, this algorithm contructs saliency weighted normal vector field and uses graph-cut for implicit surface evolution.

Mean shift is used for the explicit surface evolution. The other steps of those two algorithms are identical. The following will detail the implicit and explicit surface evolution.

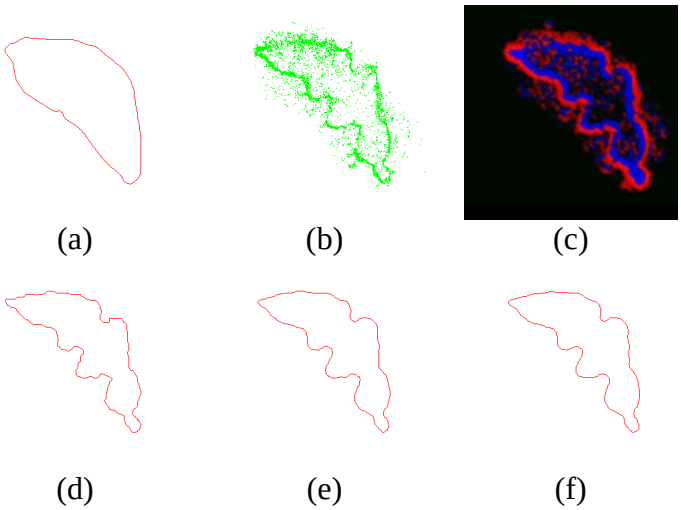


Figure 24 : A 2D slice view of the 3D reconstruction process from the dino ring data

(a) visual hull; (b) points generated by depth estimation; (c) divergence field; (d) shape estimation by graph-cut; (e) shape estimation by graph-cut after another iteration; (f) refined shape estimation by explicit surface evolution.

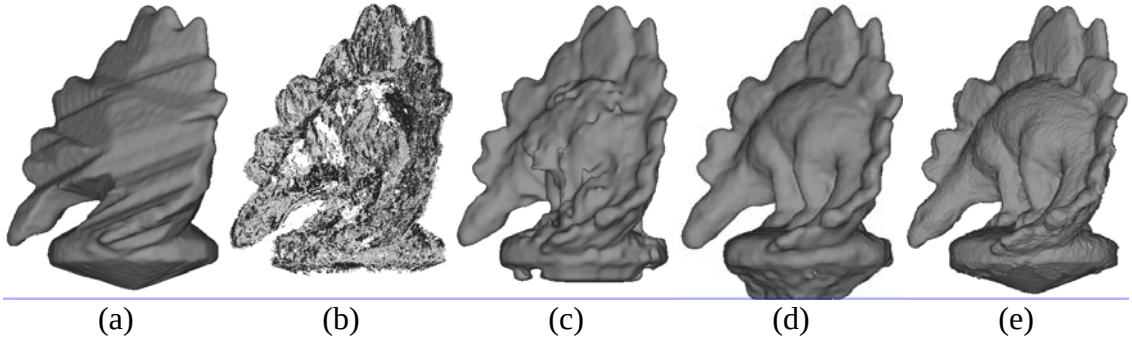
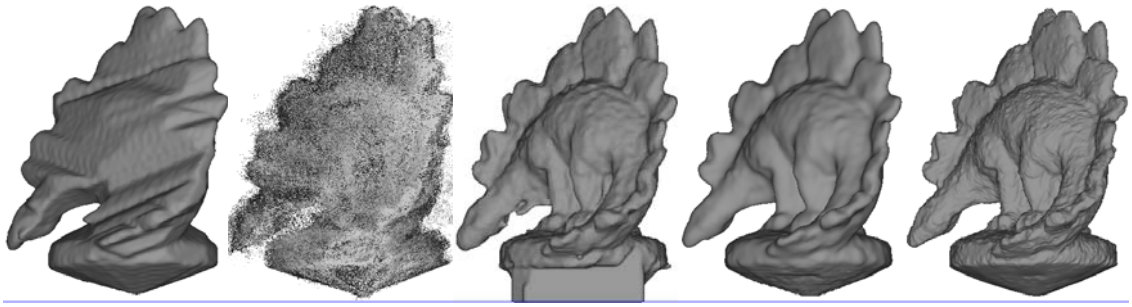


Figure 25 : 3D view of the reconstruction process of the dino sparse ring dataset of [multiv]

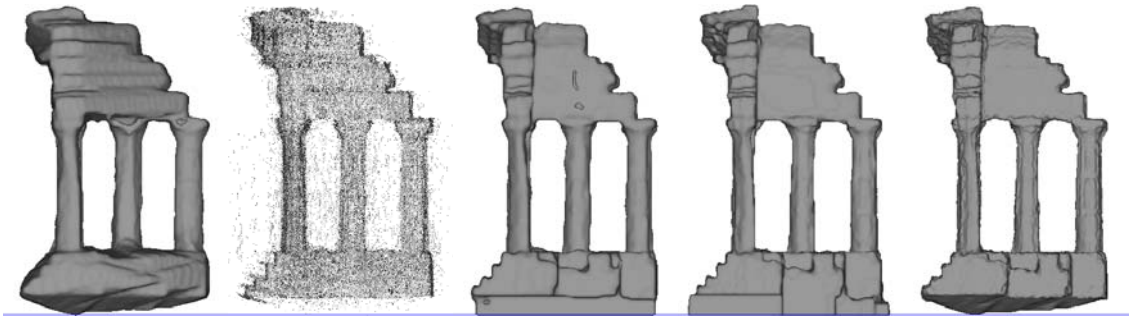
From left to right: visual hull; 3D points generated by depth estimation; shape estimation by graph-cut; shape estimation by graph-cut after another iteration; refined shape estimation by explicit surface evolution.



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

Figure 26 : 3D view of the reconstruction process of the dino ring dataset of [multiv]

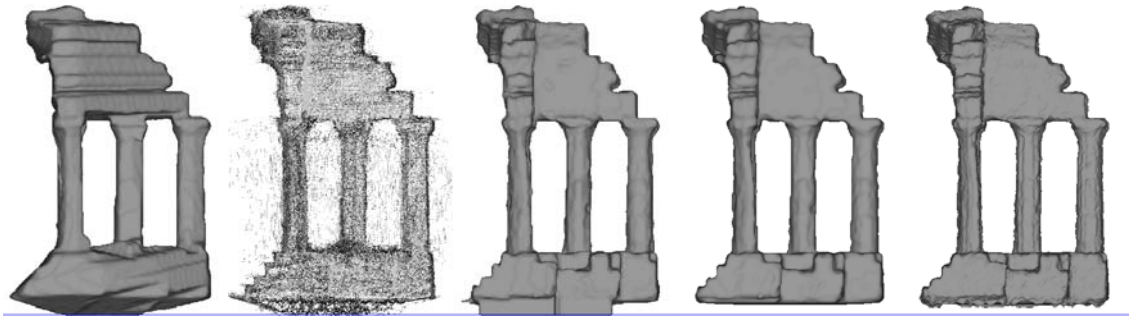
From left to right: visual hull; 3D points generated by depth estimation; shape estimation by graph-cut; shape estimation by graph-cut after another iteration; refined shape estimation by explicit surface evolution.



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

Figure 27 : 3D view of the reconstruction process of the temple sparse ring dataset of [multiv]

From left to right: visual hull; 3D points generated by depth estimation; shape estimation by graph-cut; shape estimation by graph-cut after another iteration; refined shape estimation by explicit surface evolution.



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

Figure 28 : 3D view of the reconstruction process of the temple ring dataset of [multiv]

From left to right: visual hull; 3D points generated by depth estimation; shape estimation by graph-cut; shape estimation by graph-cut after another iteration; refined shape estimation by explicit surface evolution.

4.3.1. Saliency Weighted Normal Vector Field Construction

There are two main steps in the data fusion phase. First, a saliency weighted normal vector field is constructed based on the 3D points generated by the above depth estimation step. Next, a watertight 3D surface is extracted from the saliency weighted normal vector field by energy minimization. The saliency weighted normal vector field is constructed by the following three steps: 1) saliency field construction by anisotropic kernel density estimation; 2) normal estimation and consistent normal orientation propagation; 3) volumetric saliency weighted normal vector field construction.

We will detail the three steps of saliency weighted vector field construction in this section. The 3D surface extraction will be discussed in the later sections.

4.3.1.1. Saliency field construction by anisotropic kernel density estimation

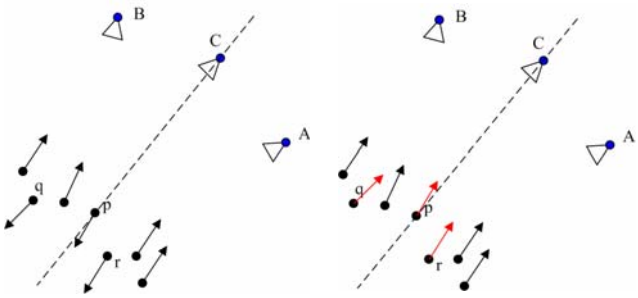
We use the term saliency to represent the likelihood the unknown surface passes through a certain part of 3D space. We propose to employ parzen-window based nonparametric density estimation method to compute the saliency of each point (see 3.2.1).

4.3.1.2. Normal estimation and consistent normal orientation propagation

Given the 3D point clouds, we can estimate the normal vector at each point based on the Principle Component Analysis (PCA) algorithm [hugu92]. Normal vectors estimated by the PCA algorithm however has an ambiguity of 180 degree so might not be consistently oriented. An orientation propagation is often needed to ensure the consistent orientation of the normal vectors. One way to do this is to first build a graph with each point as a node and the weights of edges between the adjacent points are defined as $1-\|n_1 \cdot n_2\|$, where n_1 and n_2 are the normal vectors of the two adjacent points, and then compute the minimum spanning tree (MST) from the graph using algorithms such as the Kruskal's algorithm [krus56] which finds a subset of the edges that forms a tree that includes every vertex in the graph, where the total weight of all the edges in the tree is minimized. At the termination of the algorithm, the normals are adjusted so the two neighbors in the tree have consistent normal orientation.

The above MST based normal orientation propagation approach however is not robust against noises and outliers. In the thesis, we propose to utilize external knowledge

to guide the normal orientation propagation (Figure 29). Particularly, since the point clouds are generated by depth estimation based on the image correlations between different cam-eras, for a given point p, it should be visible to its main camera in the depth estimation step (Section 4.2.2), i.e. the dot product between the normal vector of the point p and the view direction of its main camera should be negative. If not we will reverse the normal orientation at this point.



(a) (b)

Figure 29 : Camera view direction guided normal orientation propagation

(a) 3D points generated by the depth estimation based on the image correlations between the main camera C and the other adjacent cameras. Each point is shown with its normal vector estimated by the PCA algorithm. The orientation of the normal vectors at point p, q and r are not correct. (b) Based on the view direction of the main camera C, the orientation of the normal vectors at point p, q and r are reversed so that they are now opposite to the view direction of camera C.

4.3.1.3. Volumetric Saliency Weighted Normal Vector Field Construction

Once we have estimated the saliency and normal vector at each point, we will proceed to construct a volumetric saliency weighted normal vector field, from which a watertight 3D surface can be extracted by energy minimization. A volumetric grid

embedding all the 3D points is first constructed. The saliency and the normal vector of each point are then propagated to its adjacent grid nodes (Figure 30). Specifically, the saliency S_g of a grid node g is computed as the weighted summation of the saliency of its adjacent points :

$$S_g = \sum_i C_{p_i} S_{p_i} \qquad 4.2$$

The weight C_{p_i} is calculated using the aforementioned parzen-window kernel function (Equation (3.6)) based on the anisotropic Mahalanobis distance (Equation (3.10)) between the grid node g and point p_i. Since the kernel we used (e.g. truncated Gaussian kernel) has finite support here, only a finite number of points p_i $(i = 1, \ldots, n)$ within the kernel radius has non-zero weights C_{p_i}. The normal vector $\vec{N}_g$ at grid node g is calculated similarly as the weighted summation of the normal vector $\vec{N}_{p_i}$ of its adjacent points :

$$\vec{N}_g = \sum_i C_{p_i} S_{p_i} \vec{N}_{p_i} \qquad 4.3$$

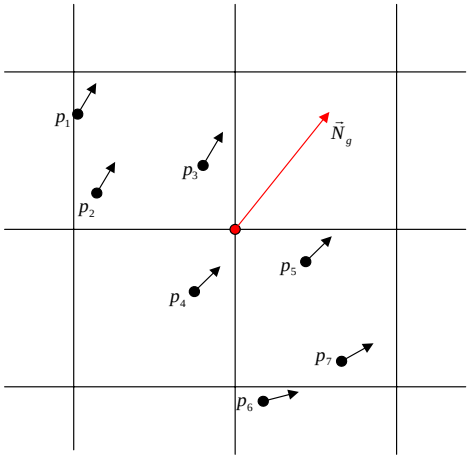
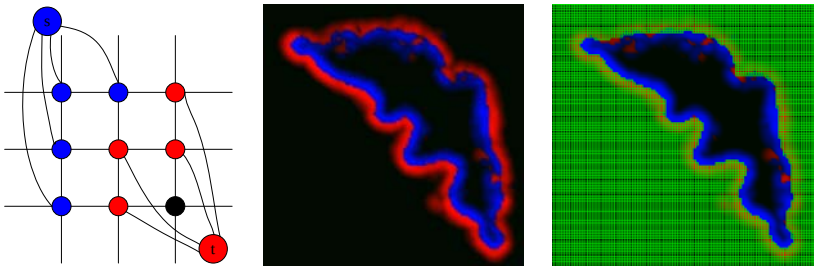


Figure 30 : Construction of the volumetric saliency weighted normal vector field

The normal vector at grid node *g* is calculated as the weighted sum of the normal vectors of its adjacent points.



(a) (b) (c)

Figure 31 : Shape estimation by graph cut

(a) Graph construction for minimization of Equation (4.8): neighboring nodes are connected via *n*-links representing regularization cost. Nodes are also connected to the terminals vial *t*-links based on their divergence value: blue nodes have positive divergence and are connected to the source terminal *s*; red nodes have negative divergence and are connected to the sink terminal *t*; the green node has zero divergence and is not connected to either terminal. (b) A 2D slice of the divergence field for the dino ring dataset of [multiv]. (c) A 2D slice of the 3D surface (shown as non-green pixels) estimated by the graph cut algorithm.

4.3.2. 3D Shape Estimation by Graph-Cut

Once the volumetric saliency weighted normal vector field is constructed, a watertight 3D surface S can be extracted by energy minimization. We use the following energy functional as suggested by [vict07]:

$$E = E_{data} + \varepsilon E_{reg} \qquad 4.4$$

E_{data} is the data alignment term which is the inverse of the flux that enforces the surface alignment with the data orientation:

$$E_{data} = -flux(S) = -\int_S \langle N, v \rangle ds \qquad 4.5$$

where <,> is (Euclidean) dot product and N is unit normal to surface elements ds consistent with a given orientation. If vectors v is interpreted as a local speed in a stream of water then the absolute value of flux equals the volume of water passing through the hypersurface in a unit of time. The sign of flux will be determined by the orientation of the surface. E_{reg} is the area-based regularization term that maintains the regularity of the extracted surface:

$$E_{reg} = \int_S ds \qquad 4.6$$

ε is the coefficient of the regularization term E_{reg} that controls the strength of the smoothness in the energy minimization process and is related to the sampling density of the data.

As pointed out by [vict07], combining flux with area based regularization can overcome the shrinking effect of the area-based regularization and improve the

reconstruction of elongated structures, narrow protrusions, and other fine details. Based on the divergence theorem for differentiable vector fields, the integral of flux of vector field over surface *S* equals to the integral of vector field's divergence *div(v)* in the interior of *S*:

$$\int_S \langle N, v \rangle ds = \int_V div(v_p) \cdot dp \qquad 4.7$$

Where *V* is the region enclosed inside *S*. Thus *E* of Equation **(**4.5**)** is now:

$$E = \varepsilon \int_S ds - \int_V div(v_p) \cdot dp \qquad 4.8$$

Equation (4.8) can be solved efficiently using the graph cut algorithm [vict07]. A typical graph construction is shown in Fig. Figure 31(a): neighboring nodes are connected via *n*-links representing area-based regularization cost. Nodes are also connected to the terminals via one *t*-link based on their divergence value: blue nodes have positive divergence and are connected to the source terminal *s* with weight $div(v_p)$; red nodes have negative divergence and are connected to the sink terminal *t* with weight $-div(v_p)$; the black node has zero divergence and is not connected to either terminal. The weight of the *n*-link is defined as the inverse of the edge length so that the weights of severed *n*-links approximate the surface area [boyk03]. Consequently, a global minimum surface for Equation (4.8) can be found by computing a minimal *s*/*t*-cut in the constructed graph.

The theory based on the graph-cut algorithm is integral geometry and Crofton formulas.

Consider in plane R^2, $L = \{(a,b) : a \geq 0, b \in [0,2\pi]\}$ is the set of all straight lines, each line $l(a,b)$ is uniquely defined by its parameter *a* and *b*, as show in Figure 32.

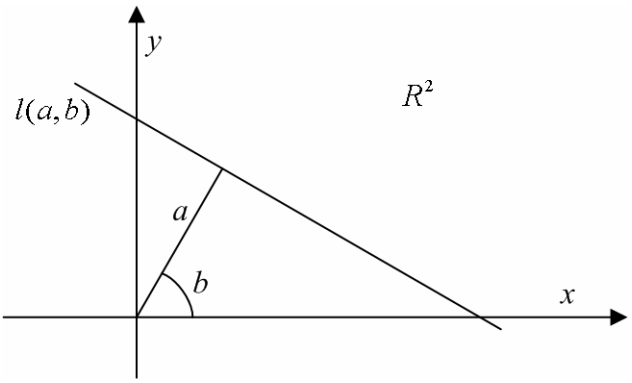


Figure 32 : Line In R^2

In R^2, each line $l(a,b)$ can be represented by two parameters *a* and *b*.

The classical Cauchy-Crofton formula demonstrates the relationship between the Euclidean length $|D|_\varepsilon$ of curve *D* in R^2 and the set of lines intersecting it.

$$\int_L n_c(l)dl = 2|D|_\varepsilon \qquad 4.9$$

Here, $n_c(l)$ is the number of times the straight line *l* intersects *D*.

Consider a weighted graph $G =< V, E >$ and a cut *C* on *G*. The cut metrics is :

$$|C|_G = \sum_{e \in C} w_e \qquad 4.10$$

w_e is the weigh of edge *e*.

If a regular grid-graph $G =< V, E >$ is embedded in plane R^2 , a closed curve *D* in R^2 is always corresponding to a cut *C* in *G*. By choose the weight of grid-graph's edges, the cut metrics $|C|_G$ will be proportional to the curve's Euclidean length $|D|_\varepsilon$.

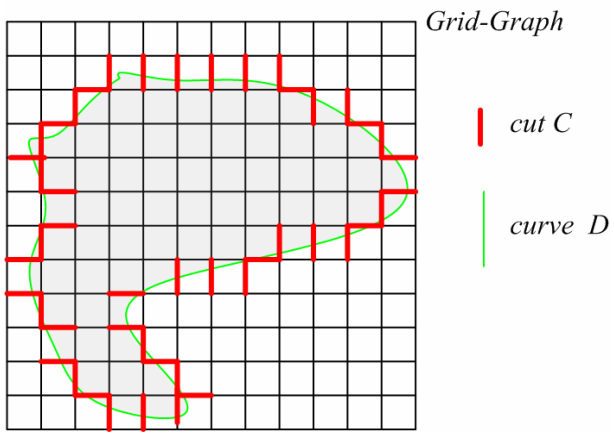


Figure 33 : Grid-graph

Grid-graph embedded in R^2 space(plane), and curve *D* is responding to the cut *C*.

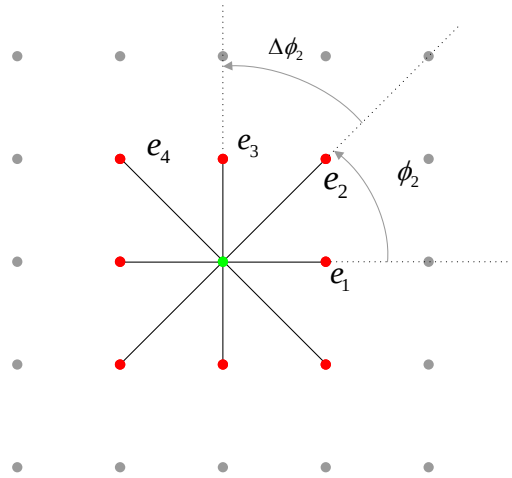


Figure 34 : Symmetric 8-neighboring system

Symmetric 8-neighboring system in 2D grid-graph. In fact, only need to consider first 4 edges because of symmetry.

[boyk03] give the detail information about how to construct such grid-graph embedded in R^n space and approximate the surface area (curve length for R^2). Figure 34 shows an example of grid-graph embedded in R^2 plane with 8-neigbhouring system.

Each node has eight symmetric neighbors, which is noted as a set $N_G = \{e_k, 1 \le k \le 8\}$. In order to make a curve's cut metrics from the grid-graph approximate its actual Euclidean length, the edge weight w_k should be :

$$w_k = \frac{\delta^2 \cdot \Delta\phi_k}{2 \cdot |e_k|} \qquad 4.11$$

Here, δ is the cell size of the grid, $\Delta\phi_k$ is the angular differences between the nearest families of edge lines, e.g, $\Delta\phi_2 = \phi_3 - \phi_2$; $|e_k|$ is the edge length. For grid-graph with symmetric neighboring system, δ, $\Delta\phi_k$ are fixed, as a result, the edge weight is inverse proportional to edge length. In this thesis, we just simple choose edge weight as :

$$w_k = \frac{1}{|e_k|} \qquad 4.12$$

It is because there is a weight factor ε in equation (4.8). To minimize equation (9), it's not necessary to calculate exactly surface area, but just choose an appropriate value of λ, which depends on the data's sampling density and the calculation of flux. To datasets in this thesis, we choose it in range between 0.02 and 0.07 for grid-graph with 26-neighboring system in R^3 space.

4.3.3. Explicit Surface Evolution

The shape estimated by the graph cut algorithm can be further improved by an explicit surface evolution step guided by the following partial differential equation (PDE):

$$\begin{cases} \dfrac{\partial S}{\partial t} = F\vec{N} \\ F(x) = \vec{m}(x) \cdot \vec{N} \\ \vec{m}(x) = \dfrac{\sum_{i=1}^{n} g(\| x - x_i \|^2) x_i}{\sum_{i=1}^{n} g(\| x - x_i \|^2)} - x \\ g(x) = k'(x) \end{cases} \qquad 4.13$$

Here S is the 3D surface, t is the time parameter, F is the speed function. $\vec{N}$ is the surface normal vector. $\vec{m}(x)$ is mean shift vector proposed in [coma02] and is proportional to $\nabla f(x)$, the gradient of the kernel density function $f(x)$ of Equation (3.6), i.e. the mean shift vector $\vec{m}(x)$ is in the gradient ascent direction and will always point to the direction of the maximal increase in the kernel density. Since the surface evolves along its normal direction (Figure 35), the speed function F is defined as the inner product between the mean shift vector $\vec{m}(x)$ and the normal vector $\vec{N}$. Note that the speed function F is dynamically calculated at each time step, thus the surface evolution is not be limited by the grid resolution.

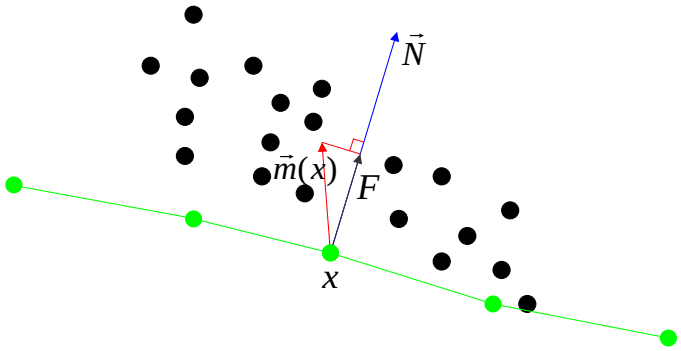


Figure 35 : Explicit surface evolution.

The surface (shown in green) will move along its normal vector with speed function F which is the inner product between the mean shift vector $\vec{m}(x)$ and the normal vector.

4.3.4. Benchmark Data Evaluation

We had applied this algorithm to the four benchmark datasets: temple ring, temple sparse ring, dino ring, and dino sparse ring from [multiv]. Table 3 shows the running time and the reconstruction accuracy obtained from the evaluation site [multiv]. The running time is based on a Pentium D Desktop PC with CPU 2.66GHz, 2GB RAM. Figure 36 are the 3D rendering of our final reconstruction results copied from the evaluation website. Our result is listed under the name “ECCV_216”.

Figure 36 : Temple and Dino 3D illustration

3D rendering of our reconstruction results running on the four Benchmark datasets of [multiv]. From top to bottom: temple ring, temple sparse ring, dino ring, and dino sparse ring.

Dataset	Running time (minutes: seconds)	# of input images	accuracy
Temple ring	33:50	47	99.5%
Temple Sparse Ring	29:15	16	96.8%
Dino ring	34:10	48	99.5%
Dino sparse ring	30:50	16	97.8%

Table 3 : Running time and reconstruction accuracy (Algorithm with graph-cut)

5 Summary

5.1. Conclusion

In this dissertation thesis, three problems related to 3D shape modeling are addressed, extraction iso-surface from volumetric dataset, elimination noisy points from points clouds and iterative surface evolution for multi-view stereo. They are focus on separate issues on 3D shape modeling but also close related to each other. The basic idea beneath them is data transformation. For cleaned points cloud, tagging algorithm can transform it to the volumetric data; for multi-view stereo, we can use line search to generate initial noisy points cloud, which can further be transformed to volumetric data via our proposed algorithm; in iteration of proposed multi-view stereo algorithm, volumetric data also need to be transformed to points cloud for new points searching.

The new region-growing based iso-surface extraction algorithm can significantly improve the original Marching Triangles algorithm. Besides using the Delaunay face property to check for existing triangulation, local iso-surface information is also checked by the new Normal Consistency Constraints to ensure the intersection of the Delaunay sphere and iso-surface is a topological disk. As can be seen from the examples, the new algorithm is very robust for datasets of complex topology and geometry. In addition, by employing a prioritized seeding initialization scheme, our algorithm can generate high-quality semi-regular meshes with different level-of-details and can preserve sharp features.

In this thesis, we propose a nonparametric approach for noisy point data preprocessing. In particular, we proposed an anisotropic kernel based nonparametric density estimation for outlier removal, and a hill-climbing line search approach for projecting data points onto the real surface boundary. Our method is very robust with highly noisy dataset, as can be seen from the examples shown. In addition, our approach is very computationally efficient. For example, it only takes 3 to 4 minutes in a middle-range desktop PC to process (outlier removal and point projection) the mult-view stereo point data (Figure 19) which contains more than 600,000 points. The majority of the running time actually spends on computing the density. The line search based projection is done in less than 10 seconds for the whole dataset.

For the most challenging multi-view stereo problem, we propose an iterative surface evolution algorithm for 3D shape reconstruction. The novel iterative refinements between image correlations based 3d depth estimation and surface evolution based shape estimation can significantly reduce the computational time and improve the accuracy of the final reconstructed surface. The benchmark evaluation results are comparable with the state-of-art methods.

In summery, those three algorithm address three close related issue in 3D shape modeling, and they work together to provide a workable way for translating most of the raw data from real world to the 3D model which can be understood and manipulated by computer in virtual world.

5.2. Future Work

Our work is an attempt to address the 3D shape modeling which bridge the gap between computer's virtual world and human beings' real world. We try to provide a workable and reliable way to help the translation. There are still lots of further improvement left.

For region growing iso-surface extraction, we would like to extend the algorithm to parallel seed initialization and out-of-core computation which will further improve its performance.

For nonparametric noisy points preprocessing, the main limitation is that currently a fixed bandwidth of the kernel is used across the whole dataset, which can cause some of the "good" data being removed during the outlier removal process (as can be seen in Figure 17 (e)). We would like to develop a scheme that can automatically select the optimal kernel bandwidth based on the local neighborhood. We would also like to extend our approach so that it can handle hole fillings automatically.

In multi-view stereo, currently our method utilizes the visual hull for initial estimation. This requires image segmentation that may be difficult for some images. We would like to relax this requirement in the future. This might be possible since our algorithm uses the iterative refinement which should be able to start from any coarse shape such as a bounding box or a convex hull.

Reference

[akko01] S. Akkouche and E. Galin, Adaptive Implicit Surface Polygonization using Marching Triangles, Computer Graphic Forum, 2001, volume 20, 2, 67-80

[amen04] Nina Amenta, Yong Joo Kil: Defining point-set surfaces. ACM Trans. Graph. 23(3): 264-270 (2004)

[amen98] N. Amenta, M. Bern, and D. Eppstein, The crust and the –skeleton: Combinational curve reconstruction, 14th ACM Symposium on Computational Geometry, 1998.

[amen98a] N. Amenta, M. Bern, and M. Kamvysselis, A new voronoi-based surface reconstruction algorithm, in SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, (New York, NY, USA), pp. 415-421, ACM Press, 1998.

[amen99] N. Amenta and M. Bern, Surface reconstruction by voronoi filtering, Discrete and Comput. Geometry, vol. 22, pp. 481-504, 1999.

[baja95] C. L. Bajaj, F. Bernardini, and G. Xu, Automatic reconstruction of surfaces and scalar fields from 3d scans, in SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, (New York, NY, USA), pp. 109-118, ACM Press, 1995.

[bois84] J. D. Boissonnat, Geometric structures for three dimensional shape reconstruction, ACM Trans. Graphics 3, pp. 266-286, 1984.

[boyk03] Yuri Boykov, Vladimir Kolmogorov: Computing Geodesics and Minimal Surfaces via Graph Cuts. ICCV 2003: 26-33.

[brad08] Bradley, D., Boubekeur, T., Heidrich, W.: Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. (2008).

[camp08] Neill Campbell, George Vogiatzis, Carlos Hernández and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. ECCV 2008, pages 766-779.

[carr01] 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, (New York, NY, USA), pp. 67-76,

ACM Press, 2001.

[case96] Vicent Caselles, Ron Kimmel, Guillermo Sapiro, Catalina Sbert: Three Dimensional Object Modeling via Minimal Surfaces. ECCV (1) 1996: 97-106.

[cign00] Paulo Cignoni and Fabio Ganovelli and Claudio Montani and Roberto Scopigno, Reconstruction of topologically correct and adaptive trilinear isosurfaces, Computers and Graphics, 2000, volume 22, 1, 312

[coma02] Comaniciu, Dorin, Peter Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence. 24 (5): 603-619. 2002.

[curl96] B. Curless and M. Levoy, A volumetric method for building complex models from range images, in SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, (New York, NY, USA), pp. 303{312, ACM Press, 1996.

[duan04] Y. Duan, L. Yang, H. Qin, and D. Samaras. Shape reconstruction from 3D and 2D data using PDE-based deformable surfaces. In ECCV, vol. 3, pp. 238-251, 2004.

[duda00] Richard O. Duda Peter E. Hart David G. Stork, Pattern Classification, October 2000, Wiley-Interscience; 2 edition.

[edel98] H. Edelsbrunner, Shape reconstruction with delaunay complex, in LATIN '98: Proceedings of the Third Latin American Symposium on Theoretical Informatics, (London, UK), pp. 119-132, Springer-Verlag, 1998.

[faug98] O. Faugeras and R. Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. IEEE Trans. on Image Processing, 7(3):336-344, 1998.

[furu07] Furukawa, Y., Pons, J. Accurate, dense, and robust multi-view stereopsis. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. (2007).

[gibs98] S. Gibson, Using Distance Maps for Accurate Surface Representation in Sampled Volumes, IEEE Symposium on Volume Visualization, 1998, 23-30

[goes06] Goesele, M., Curless, B., Seitz, S. Multi-view stereo revisited. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. (2006).

[goes07] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, Steven M. Seitz. Multi-View Stereo for Community Photo Collections, Proceedings of ICCV 2007, Rio de Janeiro, Brasil, October 14-20, 2007.

[habb07] Habbecke, M., Kobbelt, L.: A surface-growing approach to multi-view stereo reconstruction. In: Proc. IEEE Conf. on Computer Vision and Pattern

Recognition (2007).

[hama03] C. Co and B. Hamann and K. Joy, Iso-splatting: A point-based alternative to isosurface visualization, In 11th Pacific Conference on Computer Graphics and Applications (PG?3), 2003, 325-334

[heck97] M. Garland and P. S. Heckbert, Surface simplification using quadric error metrics, SIGGRAPH 97 Conference Proceddings, 1997, 209-216.

[hern02] Hernandez Esteban, C. Schmitt, F, Multi-stereo 3D object reconstruction, Proceedings of 3D Data Processing Visualization and Transmission, 2002. 159- 166.

[hern04] C. Hernandez and F. Schmitt. Silhouette and stereo fusion for 3D object modeling. CVIU, 96(3):367-392, 2004.

[hilt96] J. I. A. Hilton and A. Stoddart and T. Windeatt, Marching triangles: range image fusion for complex object modeling, International Conference on Image Processing, 1996

[hilt98] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt, Implicit surface-based geometric fusion, Comput. Vis. Image Underst., vol. 69, no. 3, pp. 273-291, 1998.

[hopp92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Surface reconstruction from unorganized points, in SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, (New York, NY, USA), pp. 71-78, ACM Press, 1992.

[hoppe93] H. Hoppe and T. Derose and T. Duchamp and J. McDonald and W. Stuetzle, Mesh optimization, SIGGRAPH 93 Conference Proceedings, 1993, 19-26.

[horn06] Hornung, A., Kobbelt, L. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In: Proc. IEEE Conf. On Computer Vision and Pattern Recognition. (2006).

[hugu92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John Alan McDonald, Werner Stuetzle: Surface reconstruction from unorganized points. SIGGRAPH 1992: 71-78.

[jim05] H. Jin, S. Soatto, and A. Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. IJCV, 63(3):175-189, 2005.

[jin03] H. Jin, S. Soatto, and A. Yezzi. Multi-view stereo beyond lambert. In CVPR, vol. 1, pp. 171-178, 2003.

[ju02] T. Ju and F. Losasso and S. Schaefer AND J. Warren, Dual contouring of hermite data, ACM Transactions on Graphics, 2002, volume 21, 3, 339-346, July

[kark01] T. Karkanis and A.J. Stewart, Curvature-dependent triangulation of implicit surfaces, IEEE Computer Graphics and Applications, 2001, volume 21, 2, 60 - 69

[kobbelt01] L. P. Kobbelt and M. Botsch and U. Schwanecke and H.-P. Seidel, Feature sensitive surface extraction from volume data, Computer Graphics (Proceedings of SIGGRAPH 2001), 57-66, 2001

[kolm02] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In ECCV, vol. III, pp. 82-96, 2002.

[krus56] Joseph. B. Kruskal: On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In: Proceedings of the American Mathematical Society, Vol 7, No. 1 (Feb, 1956), pp. 48–50.

[laur94] A. Laurentini. The visual hull concept for silhouette-based image understanding. IEEE Trans. Pattern Analysis and Machine Intelligence. Pages: 150-162, 1994.

[levi03] D. Levin. 2003. Mesh-independent surface interpolation. In Geometric Modeling for Scientific Visualization, G. Brunnett, B. Hamann, K. Mueller, and L. Linsen, Eds. Springer-Verlag.

[lorensen87] W. E. Lorensen, H. E. Cline, Marching Cubes: A high resolution 3D surface construction algorithm, Computer Graphics (Proceedings of SIGGRAPH 1987) 21 (1987) 163-169.

[maxi05] Maxime Lhuillier, Long Quan. A Quasi-Dense Approach to Surface Reconstruction from Uncalibrated Images. IEEE Trans. Pattern Anal. Mach. Intell. 27(3): 418-433 (2005).

[medi00] G. Medioni, M.-S. Lee, and C.-K. Tang, A computational framework for segmentation and grouping. Elsevier, 2000.

[merr07] Merrell, P., Akbarzadeh, A., Wang, L., Mordohai, P., Frahm, J.-M., Yang, R.,Nister, D., Pollefeys, M.: Real-time visibility-based fusion of depth maps. In: Proc. 11th Intl. Conf. on Computer Vision. (2007).

[moller98] T. Moller and K. Muller and Y. Kurzion and Raghu Machiraju and Roni Yagel, Design of accurate and smooth filters for function and derivative reconstruction, In Proceedings of IEEE Symposium on Volume Visualization, 1998, 143-51

[montani94a] C. Montani and R. Scateni and R. Scopigno, A modified look-up table for implicit disambiguation of Marching Cubes, The Visual Computer, 6, volume 10, 353-355, 1994

[multiv] The multi-view stereo evaluation web site at

http://vision.middlebury.edu/mview/.

[nielson03] Gregory M. Nielson , On Marching Cubes, IEEE Transactions on Visualization and Computer Graphics 9(3), 2003, 283 297.

[nielson04] Gregory M. Nielson, Dual Marching Cubes, 15th IEEE Visualization 2004 (VIS'04), 2004, 489-496

[nielson91] G. Nielson and B. Hamann, The asymptotic decider: resolving the ambiguity in marching cubes, IEEE Visualization, 1991, 83-91

[ohta02d] Y. Ohtake and A. Belyaev, Dual/primal optimization of polygonized implicit surfaces with sharp features, Journal of Computing and Information Science in Engineering, 2002, volume 2, 23-45

[ohta03] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H. Seidel, Multi-level partition of unity implicits, ACM Trans. Graph, 2003, Pages: 463-470.

[ouh05] Chien-Chang Ho and Fu-Che Wu and Bing-Yu Chen and Yung-Yu Chuang and and Ming Ouhyoung", Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data, Computer Graphics Forum, Vol 24, No 3, pp 537-545, 2005, 537 545.

[quan07] Long Quan, Jingdong Wang, Ping Tan, Lu Yuan: Image-Based Modeling by Joint Segmentation. International Journal of Computer Vision 75(1): 135-150 (2007)

[rodr04] Bruno Rodrigues de Araujo and Joaquim Armando Pires Jorge, Curvature Dependent Polygonization of Implicit Surfaces, Computer Graphics and Image Processing, XVII Brazilian Symposium on (SIBGRAPI'04), 2004, 266-273

[seit06] Steve Seitz, Brian Curless, James Diebel, Daniel Scharstein, Richard Szeliski, A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms, CVPR 2006, vol. 1, pages 519-526.

[sinh05] S. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In ICCV, pp. 349-356, 2005.

[soat03] S. Soatto, A. Yezzi, and H. Jin. Tales of shape and radiance in multiview stereo. In ICCV, pp. 974-981, 2003.

[vict07] Victor S. Lempitsky, Yuri Boykov: Global Optimization for Shape Fitting. CVPR 2007

[vogi05] Vogiatzis, G., Torr, P., Cipolla, R. Multi-view stereo via volumetric graph-cuts. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. (2005).

[vogi07] Vogiatzis, G., Hernandez, C., Torr, P.H.S., Cipolla, R. Multi-view stereo via volumetric graph-cuts and occlusion robust photo-consistency. IEEE Trans. Pattern Anal. Mach. Intell. 29(12) (2007).

[wang04] Yang Wang, Xiaolei Huang, Chan-Su Lee, Song Zhangand Zhiguo Li, Dimitris Samaras, Dimitris Metaxas, Ahmed Elgammal, and Peisen Huang. High resolution acquisition, learning and transfer of dynamic 3d facial expressions. Computer Graphics Forum, 23(3):677-686, 2004.

[warr04] S. Schaefer and J. Warren, Dual Marching Cubes: Primal Contouring of Dual Grids, Proceedings of Pacific Graphics 2004, 2004, 70-76

[whit97] R. Whitaker, A level set approach to 3D reconstruction from range data, International journal of Computer Vision, 1997.

[wilh92] Jane Wilhelms and Allen Van Gelder, Octrees for faster isosurface generation, ACM Trans. on Graphics, 1992, volume 11, 3, 201-27

[wyvi88] G. Wyvill and C. McPheeters and B. Wyvill, Data structure for soft objects, The Visual Computer, 1988, volume 2, 4, 227-34

[xie03] H. Xie, J. Wang, J. Hua, H. Qin, and A. Kaufman, Piecewise c1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. IEEE Visualization 2003, 91-98.

[yasu06] Yasutaka Furukawa and Jean Ponce. Carved Visual Hulls for Image-Based Modeling, volume 1, pp. 564-577. ECCV 2006.

[zhao00] H.-K. Zhao, S. Osher, B. Merriman, and M. Kang, Implicit and non-parametric shape reconstruction from unorganized data using a variational level set method, Computer Vision and Image Understanding, vol. 80, pp. 295-319, 2000.

[zhao01] H. Zhao, S. Osher, and R. Fedkiw, Fast surface reconstruction using the level set method, in VLSM '01: Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01), (Washington, DC, USA), p. 194, IEEE Computer Society, 2001.

VITA

Yongjian Xi was born on October 28, 1971 in Guiyang, China. He received his Bachelor degree in Electrical Engineering in 1994, and Master of Engineering in Electrical Engineering in 1997, both in Tsinghua University, Beijing, China. He received his Master of Science degree with major in Computer Science in Polytechnic University, New York, in 2001. He was admitted by the Department of Computer Science, University of Missouri-Columbia in the summer of 2004 and began to pursue his PhD degree in major of Computer Science from then on. Yongjian Xi has been working with Dr. Ye Duan since 2004 as a research assistant. His research interests are 3D shape modeling and computational visualization.

Publication

Journals

1. Yongjian Xi and Ye Duan, “An Iterative Surface Evolution Algorithm for Multivew Stereo”, EURASIP Journal on Image and Video Processing 2010.

2. Yongjian Xi, Ye Duan, Hongkai Zhao, “A Nonparametric Approach for Noisy Point Data Preprocessing”, International Journal of CAD/CAM volume 9, 2010.

3. Yongjian Xi and Ye Duan, “A Region-Growing Based Iso-Surface Extraction Algorithm”, Journal of Computer & Graphics, 2009.

4. Ye Duan and Yongjian Xi, “Thalamus Segmentation from Diffusion Tensor Magnetic Resonance Imaging”, International Journal of Biomedical Imaging, 2007.

5. Greg Heckenberg, Yongjian Xi, Ye Duan, and Jing Hua. "Brain Structure Segmentation from MRI by Geometric Surface Flow". International Journal of Biomedical Imaging, Vol. 2006, pp. 1 - 6, 2006.

Conferences

1. Yongjian Xi, Ye Duan, “A Integrated Depth Fusion Algorithm for Multi-View Stereo”, European Conference on Computer Vision, 2010, submitted.

2. Yongjian Xi, Ye Duan, Hongkai Zhao, “A Nonparametric Approach for Noisy Point Data Preprocessing”, The 11th IEEE International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics 2009), 2009.

3. Yongjian Xi, Ye Duan, “A Region-Growing Based Iso-Surface Extraction Algorithm”, The 10th IEEE International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics 2007), Beijing, China, October 2007.

4. Guangyu Zou, Yongjian Xi, Greg Heckenberg, Ye Duan, Jing Hua, and Xianfeng Gu, "Integrated Modeling of PET and DTI Information based on Conformal Brain Mapping". In Proceedings of SPIE: Medical Imaging, 2006.

5. Greg Heckenberg, Yongjian Xi, Ye Duan, Jing Hua, and Otto Muzik, "Thalamus Segmentation from MRI Images by Lagrangian Surface Flow". In Proceedings of 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (IEEE-EMBC05), September 2005.

6. Yongjian Xi, Greg Heckenberg, Ye Duan, Jing Hua, "Iso-surface extraction by Front Propagation ", Proceedings of Pacific Graphics 2005, Macao, China, 2005.

7. Yongjian Xi, Greg Heckenberg, Ye Duan and Hongkai Zhao, "A New Modeling-Based Algorithm for Implicit Surface Polygonization", Proceedings of Vision Geometry XIII, SPIE 2005, January 2005, San Jose, CA.