Western Graduate&PostdoctoralStudies

Western University
Scholarship@Western

Electronic Thesis and Dissertation Repository

8-18-2011 12:00 AM

Surface Reconstruction from Unorganized Point Cloud Data via Progressive Local Mesh Matching

Ji Ma, *The University of Western Ontario*

Supervisor: Hsi-Yung Feng, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Mechanical and Materials Engineering

© Ji Ma 2011

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Computer-Aided Engineering and Design Commons

Recommended Citation

Ma, Ji, "Surface Reconstruction from Unorganized Point Cloud Data via Progressive Local Mesh Matching" (2011). *Electronic Thesis and Dissertation Repository*. 231.
https://ir.lib.uwo.ca/etd/231

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

SURFACE RECONSTRUCTION FROM UNORGANIZED POINT CLOUD DATA VIA PROGRESSIVE LOCAL MESH MATCHING

(Spine title: Surface Reconstruction via Progressive Local Mesh Matching)

(Thesis format: Monograph)

by

Ji Ma

Graduate Program in Mechanical and Materials Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

© Ji Ma 2011

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor

Dr. Hsi-Yung Feng

Co-Supervisor

Dr. Lihui Wang

Supervisory Committee

Dr. George Knopf

Examiners

Dr. Anand Singh

Dr. Ralph Buchal

Dr. Jagath Samarabandu

Dr. Fengfeng Xi

The thesis by

Ji Ma

entitled:

Surface Reconstruction from Unorganized Point Cloud Data via Progressive Local Mesh Matching

is accepted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Date

Chair of the Thesis Examination Board

ABSTRACT

Nowadays point cloud (a set of dense discrete points) has become an emerging data format to represent 3D surface geometry due to the increasing application of 3D laser scanning systems. Converting such a discrete point representation into a continuous surface representation is known as surface reconstruction. Many computer-aided design and inspection applications demand an accurately reconstructed surface corresponding to a watertight triangle mesh passing through the scanned point cloud data. Automatic reconstruction of a watertight triangle mesh with correctly represented sharp features remains an open issue in surface reconstruction research.

This thesis presents an integrated triangle mesh processing framework for surface reconstruction based on Delaunay triangulation. It features an innovative multi-level inheritance priority queuing mechanism for seeking and updating the optimum local manifold mesh at each data point. The proposed algorithms aim at generating a watertight triangle mesh interpolating all the input points data when all the fully matched local manifold meshes (umbrellas) are found. Compared to existing reconstruction algorithms, the proposed algorithms can automatically reconstruct watertight interpolation triangle mesh without additional hole-filling or manifold post-processing. The resulting surface can effectively recover the sharp features in the scanned physical object and capture their correct topology and geometric shapes reliably. The main Umbrella Facet Matching (UFM) algorithm and its two extended algorithms are documented in detail in the thesis. The UFM algorithm accomplishes and implements the core surface reconstruction framework based on a multi-level inheritance priority queuing mechanism according to the progressive matching results of local meshes. The first extended algorithm presents a new normal vector combinatorial estimation method for point cloud data depending on local mesh matching results, which is benefit to sharp features reconstruction. The second extended algorithm addresses the sharp-feature preservation issue in surface reconstruction by the proposed normal vector cone (NVC) filtering. The effectiveness of these algorithms has been demonstrated using both simulated and real-world point cloud data sets. For each algorithm, multiple case studies are performed and analyzed to validate its performance.

Keywords: Surface reconstruction, Point cloud, Triangle mesh, Delaunay triangulation, Local mesh matching, Priority queue, Sharp feature

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my supervisor Prof. Hsi-Yung (Steve) Feng and co-supervisor Dr. Lihui Wang, for their continuous support and enthusiastic guidance over the past five years. I owe many thanks to Prof. Feng for his patience and encouragement throughout my graduate study.

I would like to thank all members of our CAD/CAM/CAI research group for their support and friendship. I would also like to extend my appreciation to members of my supervisory and thesis examination committees: Profs. George Knopf, Anand Singh, Ralph Buchal, Jagath Samarabandu and Fengfeng Xi for their serious evaluations and constructive suggestions.

Finally, I would like to thank my parents and brothers for their support, encouragement, prayer and unconditional love over the years. You have always been there for me. In particular, I would like to express my appreciation to my wife for her understanding, continuous support, kindness, and endless love.

TABLE OF CONTENTS

CERTIFICATE OF EXAMINATION ii

ABSTRACT iii

ACKNOWLEDGMENTS v

TABLE OF CONTENTS vi

LIST OF TABLES viii

LIST OF FIGURES ix

NOMENCLATURE ii

1 INTRODUCTION 1
- 1.1. Background and Motivation 1
- 1.2. Scope and Objectives 3
  - 1.2.1. Problem Definition 3
  - 1.2.2. Geometry Processing Based on Triangle Mesh 4
  - 1.2.3. Research Objectives 5
- 1.3. Thesis Structure 8

2 PREREQUISITES AND LITERATURE REVIEW 10
- 2.1. Mathematical Prerequisites 10
  - 2.1.1. Voronoi Diagram and Delaunay Triangulation 10
  - 2.1.2. Related Concepts 14
- 2.2. Existing Surface Reconstruction Approaches 18
  - 2.2.1. Implicit Surface 18
  - 2.2.2. Region Growing 21
  - 2.2.3. Delaunay-based 23
- 2.3. Outstanding Issues 26

3 WATERTIGHT SURFACE RECONSTRUCTION VIA PROGRESSIVE UMBRELLA FACET MATCHING 29
- 3.1. Introduction 29
- 3.2. Relevant Concepts 31
  - 3.2.1. Definition of the Umbrella 31
  - 3.2.2. Delaunay Triangle Clusters 32

3.3. Umbrella Facet Matching....34
3.3.1. Building an Umbrella....35
3.3.2. Priority Queuing Mechanism and the Initial Queue....36
3.3.3. Evaluation of the Matching Results....38
3.3.4. Computational Complexity....42
3.4. Implementation Results and Discussion....45
3.5. Concluding Remarks....56

4 NORMAL VECTOR ESTIMATION FOR POINT CLOUD DATA....58
4.1. Introduction....58
4.2. Reliable Neighborhood Identification....59
4.2.1. Existing Approaches....59
4.2.2. Local Mesh Neighborhood....62
4.3. Computational Procedure....65
4.4. Implementation Results and Discussion....69
4.4.1. Case Study Setup....69
4.4.2. Analysis and Comparison....73
4.4.3. Limitation....78
4.5. Concluding Remarks....81

5 NORMAL VECTOR CONE FILTERING FOR SHARP FEATURE RECONSTRUCTION....83
5.1. Introduction....83
5.2. Relevant Techniques....87
5.2.1. Feature Extraction from Point Clouds....87
5.2.2. Feature Identification from Meshes....88
5.3. Overview of the Proposed Method....89
5.4. Feature Sensitive Umbrella Update....91
5.4.1. Priority Queuing Based on Matching Results....91
5.4.2. Improved Priority Queuing for Sharp-feature Preservation....94
5.4.3. Normal Vector Cone Filtering....96
5.5. Implementation and Results....105
5.6. Concluding Remarks....117

6 CONCLUSIONS AND FUTURE WORK....119
6.1. Main Contributions....119
6.2. Future Work....125

REFERENCES....128

CURRICULUM VITAE....135

LIST OF TABLES

Table 2.1 Corresponding geometry elements in dual relationship between Voronoi diagram and Delaunay triangulation 13

Table 3.1 List of possible matching index values for facet $f(v_1, v_2, v_3)$ 40

Table 3.2 Implementation results for some public point cloud data sets 46

Table 3.3 Comparison of the reconstructed meshes-algorithms not employing hole-filling post-processing 50

Table 3.4 Comparison of the reconstructed meshes-algorithms employing hole-filling post-processing 51

Table 4.1 Normal vector estimation errors (in radians) 72

Table 5.1 Classification of point with an umbrella 99

Table 5.2 Implementation results for some public point cloud data sets 106

LIST OF FIGURES

Figure 1.1 The general pipeline of 3D data acquisition and processing 2

Figure 1.2 An example of surface reconstruction 4

Figure 1.3 Geometry processing based on triangle mesh 5

Figure 1.4 Structure of the thesis. 9

Figure 2.1 (a) Voronoi diagram and (b) Delaunay triangulation in the plane 11

Figure 2.2 The Voronoi diagram and its dual graph Delaunay triangulation 12

Figure 2.3 An example of Gabriel graph in the plane.. 14

Figure 2.4 Some examples of *k*-ball and *k*-sphere in [5] 15

Figure 2.5 Examples of homeomorphism in [4] 16

Figure 2.6 Non-manifold cases of triangle mesh in [1] 17

Figure 2.7 Interpolation and approximation of a 2D point set in [10] 19

Figure 2.8 A 2-dimensional example of BPA algorithm [24] 21

Figure 2.9 A 2-dimensional example of Delaunay-based surface reconstruction 24

Figure 2.10 A 2-dimensional example of the crust algorithm [35]................................. 25

Figure 3.1 Umbrella U_v at a point v 32

Figure 3.2 Topological types of Delaunay triangle clusters: (a) umbrella; (b) umbrella with fins; (c) umbrella with pockets; and (d) umbrella with fins and pockets. 33

Figure 3.3 Priority queuing mechanism... 36

Figure 3.4 Absolute and relative matching indices of f at v 38

Figure 3.5 Typical matching results for an umbrella facet 41

Figure 3.6 Umbrella Facet Matching algorithm flowchart 43

Figure 3.7 Detail UFM convergence process for the Mechpart data set 44

Figure 3.8 Reconstructed mesh for the Mannequin data set 47

Figure 3.9 Reconstructed mesh for the Knot data set 48

Figure 3.10 Other reconstructed meshes for the data sets in Table 3.2 49

Figure 3.11 Comparison of reconstructed mesh of the Casting Die data set 53

Figure 3.12 Convergence problem in highly non-uniform and under-sampled data 54

Figure 3.13 Shape deviations in the sharp features 54

Figure 3.14 Performance of the UFM algorithm with increasing noise 55

Figure 4.1 An example of local mesh neighbors based on umbrella matching 64

Figure 4.2 Global meshes comparison based on different local mesh neighbors 65

Figure 4.3 Normal vector estimation flowchart 68

Figure 4.4 The first type data for normal vector estimation comparison 70

Figure 4.5 The second type data for normal vector estimation comparison 70

Figure 4.6 Comparison of estimated normals for simulated data 74

Figure 4.7 Comparison of estimated normals for simulated data with sharp features 74

Figure 4.8 Comparison of estimated normals for Cubic data 76

Figure 4.9 Comparison of estimated normals for real point cloud data 78

Figure 4.10 Torus with increasing noise 79

Figure 4.11 Estimated normal errors in different noise level 80

Figure 4.12 Proposed algorithm with two-ring neighbors 81

Figure 5.1 An example with sharp features: (a) original point cloud; (b) reconstructed mesh by a general algorithm; and (c) reconstructed mesh by the proposed feature sensitive algorithm 84

Figure 5.2 Umbrella U_v at point v 89

Figure 5.3 Three stages of building a matched umbrella at point v 92

Figure 5.4 Multi-level inheritance priority queuing 93

Figure 5.5 Feature sensitive priority queuing 95

Figure 5.6 Normal vector cone at point v 96

Figure 5.7 Four different point types 100

Figure 5.8 NVC filtering flowchart 103

Figure 5.9 Comparison of reconstructed surface for Casting Die data set 108

Figure 5.10 Reconstructed surface for the data set in Table 5.2 110

Figure 5.11 Comparison of resulting meshes of Fandisk01 and Fandisk02 data set 111

Figure 5.12 Comparison of resulting meshes of Oilpmp data set 112

Figure 5.13 Comparison of resulting meshes with sharp features of some examples ... 114

Figure 5.14 Shape comparison of resulting meshes of simulated data set 115

Figure 5.15 Colour map comparison of resulting meshes of simulated data set 115

Figure 5.16 Loss of sharp features 116

Figure 6.1 Automatic watertight manifold surface reconstruction via progressive local mesh matching 120

Figure 6.2 Progressive local mesh matching mechanism 121

Figure 6.3 Comparison of estimated normal vector from: (a) weighted plane fitting algorithm; (b) the proposed algorithm in chapter 4 123

Figure 6.4 Normal vector cone filtering 124

Figure 6.5 An example of the sharp features missed 126

NOMENCLATURE

P = given a point cloud or point set

p or v = a point in P

S = original surface of a physical object

S' = reconstructed (triangle mesh) surface

$V(P)$ = Voronoi diagram of a point set P

V_p = Voronoi cell corresponding to a point p

$D(P)$ = Delaunay triangulation of a point set P

DT_v = Delaunay triangle set incident to a point v

$GT(P)$ = all Gabriel triangles in $D(P)$

$GG(P)$ = Gabriel graph of a point set P

GT_v = Gabriel triangle set incident to point v

$DT(U)_v$ = umbrella Delaunay-triangle set incident to point v

U_v = an umbrella at point v

$U(f)_v$ = triangular facet set in the umbrella at point v

$U(\bar{f})_v$ = all fully matched triangular facets in the umbrella at point v

$U(p)_v$ = circumjacent neighboring point set in the umbrella at point v

f = a triangular facet of the umbrella

$\bar{f}$ = a fully matched triangular facet of the umbrella

M_f = absolute matching index of f

$M_{f(v)}$ = relative matching index of f with respect to v

$\widetilde{U}_v$ = void matched umbrella at point v

$\bar{U}'_v$ = partially matched umbrella at point v

$\bar{U}_v$ = fully matched umbrella at point v

G_v = grade of point v with an umbrella

ΔF = topology deviation of a reconstructed triangle-mesh surface

V = number of vertices in a triangle mesh

F = number of triangles in a triangle mesh

G = genus of a physical object

$N_k(p)$ = *k*-nearest neighbors of point p

$\bar{N}$ = nominal normal of a Normal Vector Cone

α = cone angle of a Normal Vector Cone

R = limitation range of a Normal Vector Cone

$\theta_{dihedral}$ = dihedral angle of two adjacent triangles

1 INTRODUCTION

1.1. Background and Motivation

The past few decades have seen more and more applications of 3D data acquisition technologies in many disciplines. For example, in manufacturing industries, both traditional Coordinate Measuring Machine (CMM) contact measurement and emerging 3D laser scanner non-contact measurement have become the most significant 3D data acquisition applications in reverse engineering. In computer graphics community, it is often required to capture complex 3D shapes on site by portable laser scanner for computer simulation and animation. In addition, X-rays, Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) scanning are all typical data acquisition applications in medical field. In all these applications, data sources from various data acquisition devices consist of discrete sampling data, which could be further divided into different categories: unorganized data, contour data, volumetric data, range data, and so on. Converting the discrete sampling data representation of a physical object into a continuous surface of digital representation in computer is known as surface reconstruction.

If the discrete sampling data has enough resolution to represent the scanned model surface, the surface reconstruction would recover the topology and geometry of the model surface. The general pipeline of the 3D data acquisition and processing from the initial physical object in real-world to the final digital model in computer-world is shown in Figure 1.1. The first stage involves the acquisition of the discrete sample from a physical object through 3D data acquisition system. The second stage includes reconstructing basic geometric surface model from the acquired discrete sampling data. Once the geometric model is in place, various application-specific modeling and digital processing can be launched in the third stage. In all the stages of the data acquisition and processing, surface reconstruction undoubtedly stands out as the most significant and challenging task in obtaining the digital model from the physical object.

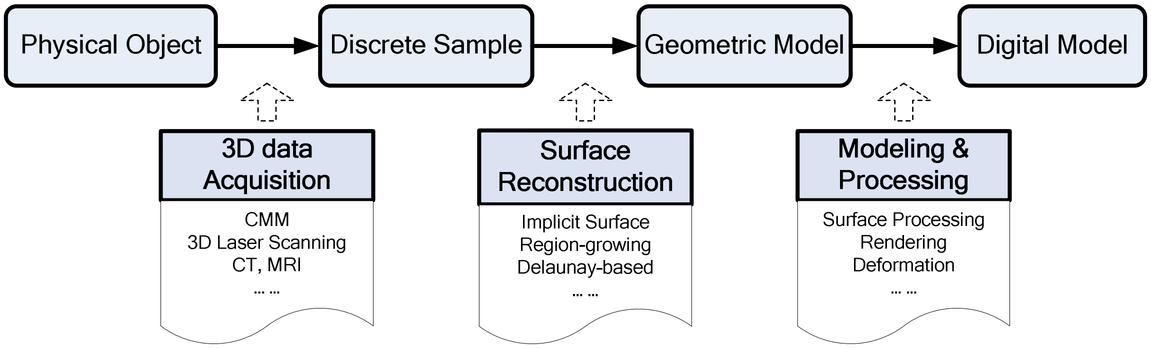


Figure 1.1 The general pipeline of 3D data acquisition and processing

In manufacturing industries, 3D data acquisition based on laser scanning system has become a standard in capturing the complicated surface geometry of physical objects for applications such as CAD/CAM/CAI. Point cloud (unorganized point set data) from 3D laser scanner is emerging as a new data format for representing surface geometry of a scanned model, which includes no more information than the coordinates of measured points in the most general sense. Meanwhile, triangle meshes have become increasingly popular in representing piecewise linear C^0 continuous surface, and are employed intensively in computer graphics and geometric processing: the sheer simplicity in concept allows for maximal flexibility and efficiency in computer processing. Triangle meshes have thus developed into a favourable alternative to traditional spline surfaces and are widely adopted in CAD/CAM/CAI applications, mesh-based numerical simulation and analysis, computer game and movie/animation production. In this context, the surface reconstruction proposed in this thesis will focus on converting the unorganized point cloud data from laser scanner into the triangle mesh surface.

1.2. Scope and Objectives

1.2.1. Problem Definition

The definition of surface reconstruction can be stated as: given an unorganized point cloud P in Euclidean space R^3, scanned from an original surface S in Euclidean space R^3, to reconstruct a triangle mesh surface S' (C^0 continuity) so that the points of P lie on or close to S' and the surface S' is topologically equivalent and geometrically close to S. Depending on the specific application, one can choose the reconstructed triangle mesh to either interpolate (pass through) all measurement points in the point cloud or approximate them within a given tolerance [1]. Main characters are listed here:

Input: the unorganized data point cloud $P \subset R^3$ from an original surface $S \subset R^3$.

Output: reconstructed triangle mesh surface S' (C^0 continuity) topologically equivalent and geometrically close to S

Choice: interpolation or approximation of the point cloud P

Figure 1.2 shows one example of surface reconstruction. Regarding surface reconstruction, the original surface S of physical object is unknown except the point cloud P scanned from original surface S: which is the ensemble of the coordinates of scanned points. The desired product of surface reconstruction is a reconstructed triangle mesh surface S' which is able to represent the correct topology with reliable approximation of the geometry of S. The reconstructed triangle mesh surface S' can be configured either as interpolation of all measurement points in point cloud P or approximation thereof with prescribed degree of tolerance.

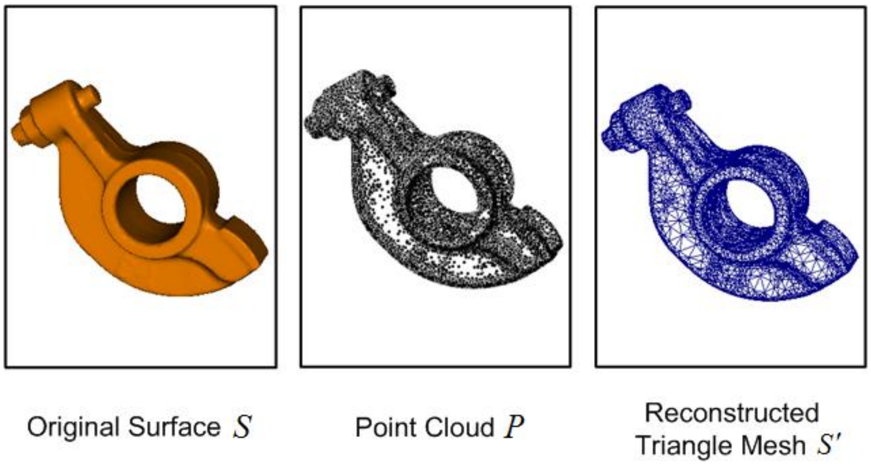


Figure 1.2 An example of surface reconstruction

1.2.2. Geometry Processing Based on Triangle Mesh

As the sole input, obtaining and optimizing point cloud data scanned from physical object is crucial for successful surface reconstruction. Point cloud data scanned from general 3D laser scanner is often the dense and noisy data, especially in some small sharp features region. Consequently, a specific pre-processing of point cloud, such as denoising and simplification of raw point cloud data, is often required as an essential step prior to surface reconstruction. Figure 1.3 depicts a general geometry processing procedure based on triangle-mesh surface from the raw input point cloud data to the final desired triangle mesh digital model. Most surface reconstruction algorithms fall into three categories: implicit surface, region growing and Delaunay-based approaches. Depending on different reconstruction approach, the reconstructed triangle meshes often require specific post-processing, such as hole-filling post-processing, for more accurate recovery of geometric surface information of scanned physical object. Mesh optimization further improves the quality of the reconstructed triangle-mesh surface for next-level geometric processing by smoothing, subdivision, remeshing, and so forth. Finally, all kinds of modeling technology based on meshes can be applied for different computer applications.

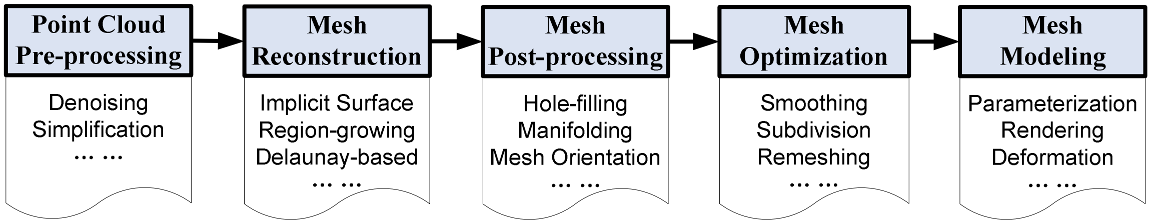


Figure 1.3 Geometry processing based on triangle mesh

1.2.3. Research Objectives

Although many surface reconstruction schemes have been proposed in the past few decades, accurate surface reconstruction remains a challenge in practice due to the sparsity, redundancy, noisiness of the acquired point cloud data and/or the non-smoothness and sharp boundaries of the original surface of physical object. Instead of one single dominant method, there are many different approaches that are currently in use depending on input properties, the output requirement, software availability, user preference, etc.

Recently in manufacturing industry, 3D data acquisition technologies based on ultra-precise 3D laser scanning system have gained popularity in computer-aided design and inspection, and become a powerful tool in capturing accurate geometry of complicated physical objects. For example, the Surveyor Laser Probes of Laser Design, Inc. can capture up to 225,000 points per second and their accuracy can achieve up to 10 μm, which provide a high-accuracy, high-speed, non-contact 3D scanning for industry. Many computer-aided design and inspection applications demand high-quality surface reconstruction corresponding to a watertight manifold triangle-mesh surface for advanced subsequent process. Furthermore, since all measurement points in a point cloud data come from a physical object in real-world, the scanned raw measurement information should be preserved for subsequent process or analysis. This requires the reconstructed triangle mesh surface passing through (interpolating) all raw measurement points in scanned unorganized point cloud data, which can still be pre-processed through the specific denoising or simplification process for point cloud data as mentioned previously. Automatic reconstruction of such triangle mesh surfaces with correctly represented sharp features remains an open research topic in surface reconstruction research.

This thesis is dedicated to watertight triangle mesh surface reconstruction with emphasis on the recovery of the sharp features. The reconstructed triangle mesh surface is manifold mesh and passes through all measurement points. In the context of this thesis, we assume the input point cloud P in Euclidean space R^3 is a low-noise, unorganized coordinate data set containing no other geometric information (such as surface normals). Unlike other organized data, such as contour data and volumetric data, unorganized point cloud data input also means that the proposed reconstruction algorithms in this thesis focus on the general surface reconstruction problems. They do not assume any additional structure information or relationship information among input point cloud data except pure 3D coordinate value of each point. The genus of the original surface S of scanned physical object is also not limited in our treatment. The objective of proposed surface reconstruction approach is to create a triangle mesh surface S' (C^0 continuity) so that all measurement points of P lie on S' and the reconstructed surface S' is a watertight manifold surface expected to capture correct topology with reliable geometric approximation. Main characters are listed here:

> **Objective:** triangle mesh surface $S' \subset R^3$ (C^0 continuity) with high quality (watertight manifold triangle mesh; passing through all measurement points in P; topologically equivalent and geometrically close to S; preserving the sharp features of S well).
>
> **Assumption:** an unorganized, low-noise point cloud $P \subset R^3$ from the original surface $S \subset R^3$ of a physical object.

Watertight surface is a close surface that bounds a solid. It could be formally defined [2] as a 2-complex embedded in Euclidean space R^3 whose underlying space is same as the boundary of the closure of a 3-manifold in Euclidean space R^3.

1.3. Thesis Structure

Relevant mathematical concepts and literature review are presented in Chapter 2. Chapter 3 introduces the core Umbrella Facet Matching (UFM) algorithm, which provides a surface reconstruction framework based on a multi-level priority queuing mechanism according to the progressive matching results of local meshes. The first extended algorithm introduced in Chapter 4 proposes a new normal vector estimation method for point cloud data from local mesh matching results. The second extended algorithm introduced in Chapter 5 addresses the sharp-feature preservation issue in surface reconstruction by an innovative normal vector cone (NVC) filter, which is an extended UFM algorithm in fact. Chapter 6 discusses main contributions and directions for future research. The structure of this thesis is shown as flow chart in Figure 1.4.

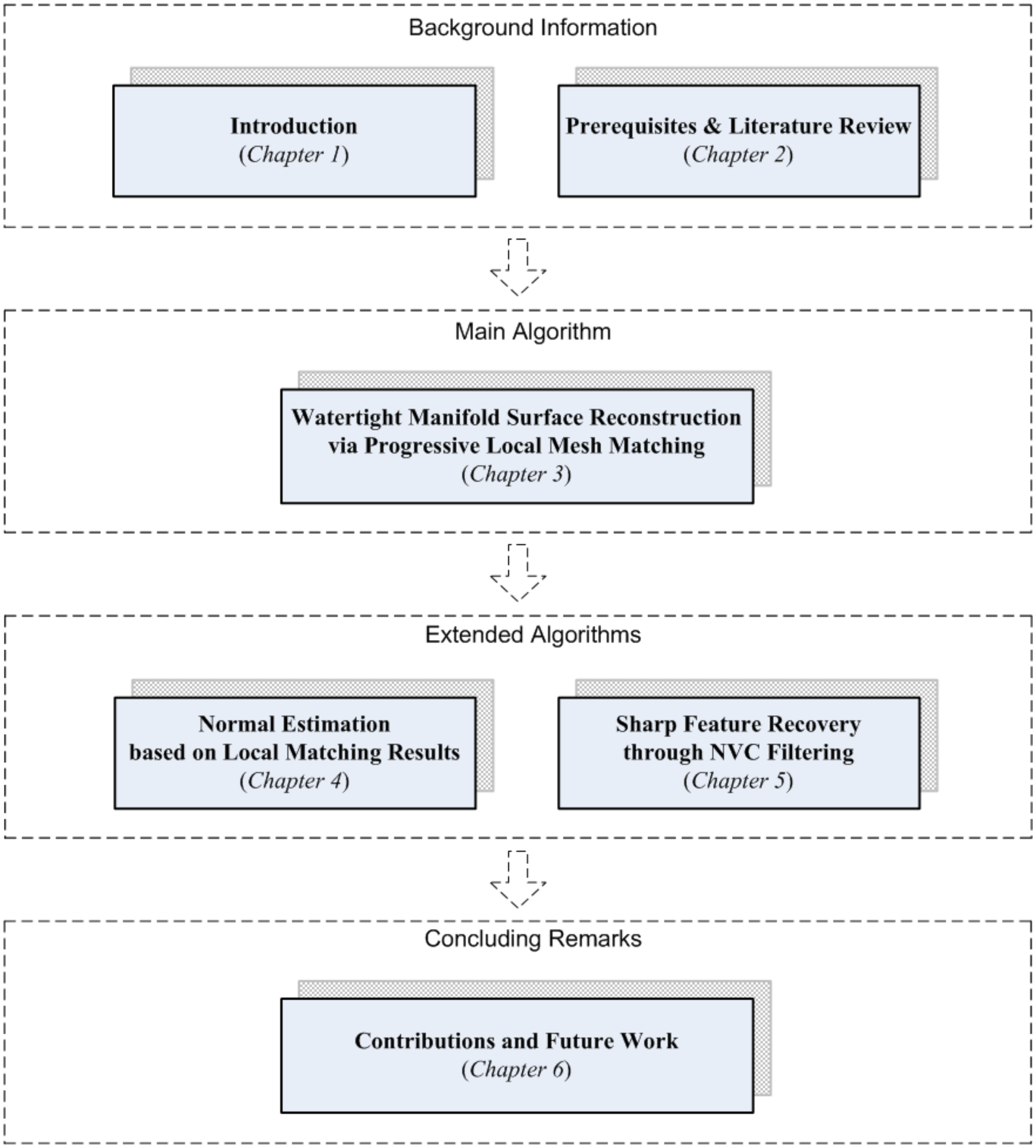

Background Information
Introduction
(Chapter 1)
Prerequisites & Literature Review
(Chapter 2)
Main Algorithm
Watertight Manifold Surface Reconstruction via Progressive Local Mesh Matching
(Chapter 3)
Extended Algorithms
Normal Estimation based on Local Matching Results
(Chapter 4)
Sharp Feature Recovery through NVC Filtering
(Chapter 5)
Concluding Remarks
Contributions and Future Work
(Chapter 6)

Figure 1.4 Structure of the thesis.

2 PREREQUISITES AND LITERATURE REVIEW

2.1. Mathematical Prerequisites

Surface reconstruction algorithms described in this thesis construct a piecewise linear approximation (triangle mesh) of the original physical object surface by approximating or interpolating the scanned point cloud data. This mathematical approximation is intended to capture the correct topology and geometric shapes of the original physical object surface. A necessary preparation is given here on some basic concepts and terminology from related mathematic disciplines, such as computational geometry and point set topology.

2.1.1. Voronoi Diagram and Delaunay Triangulation

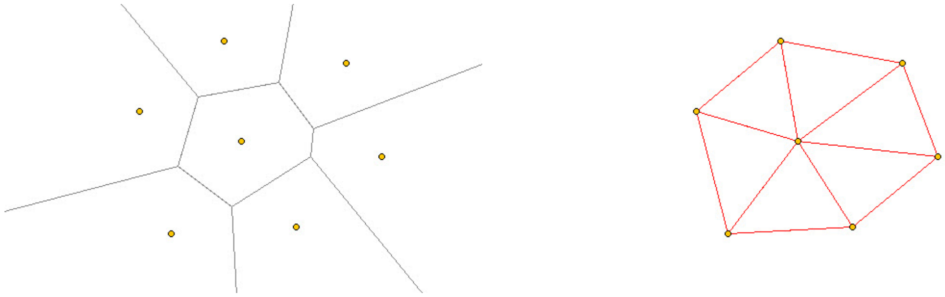
Voronoi diagram and Delaunay triangulation are essential geometric data structures that are built upon the notion of "neighbor". Delaunay triangulation is the dual graph of Voronoi diagram. For curves and surfaces in Euclidean space, their many differential properties are defined with a local neighborhood. Voronoi diagram and Delaunay triangulation can provide a powerful way to approximate the neighborhood in the discrete domain, such as the discrete points set. Some basic related concepts on them are introduced in this section.

Voronoi diagram

The Voronoi diagram $V(P)$ of a point set P is defined as a neighbourhood region decomposition of Euclidean space R^3. Every neighbourhood region is a cell, which is called Voronoi cell. Each Voronoi cell corresponds to exactly one point and contains all points of R^3 that do not have a smaller distance to any other point in point set P. The Voronoi cell corresponding to each point $p \in P$ is given as follows [3]

$$V_p = \left\{ x \in R^3 : \forall q \in P \quad \|x - p\| \leq \|x - q\| \right\} \tag{2.1}$$

Closed faces shared by two Voronoi cells are called Voronoi faces. In this cell decomposition, the rest geometric elements includes: Voronoi edges and Voronoi vertices. Voronoi objects represent all these geometric elements. The collection of all Voronoi objects creates the Voronoi diagram. A 2-dimensional example of a Voronoi diagram is shown in Figure 2.1a.



(a)

(b)

Figure 2.1 (a) Voronoi diagram and (b) Delaunay triangulation in the plane

Delaunay triangulation

The Delaunay triangulation $D(P)$ of P is a dual graph of the Voronoi diagram. Figure 2.1b shows a 2-dimensional example of a Delaunay triangulation, which is the dual of the Voronoi diagram in Figure 2.1a. In this 2-dimensional example, the Voronoi diagram is built by all perpendicular bisectors of a pair of “adjacent” points in points set

P. By connecting all point pair, the Delaunay triangulation of P is built. Corresponding to different Voronoi objects, there exist different Delaunay simplexes: Delaunay cell, Delaunay face, Delaunay edge and Delaunay vertex. Every point in P is just a Delaunay vertex and Delaunay cell is a tetrahedron in Euclidean space R^3. More details on their dual relationships are shown in the following Table 2.1.

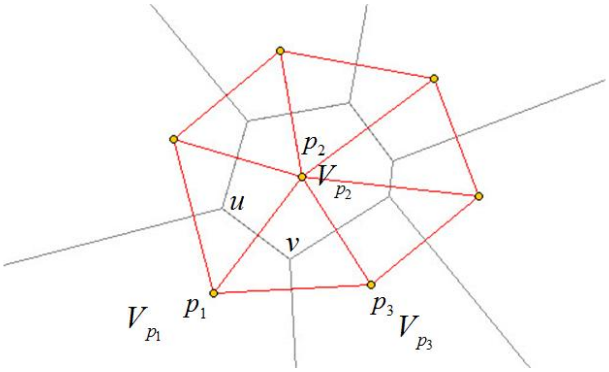


Figure 2.2 The Voronoi diagram and its dual graph Delaunay triangulation

Another two-dimensional example of the dual relationship between Voronoi diagram and Delaunay triangulation is illustrated in Figure 2.2. u and v are two Voronoi vertices and uv is a Voronoi edge. Some of the Voronoi cells (polygonal cells by gray lines in Figure 2.2) may be unbounded with unbounded edges. It is inherent that a Voronoi cell V_p is unbounded if and only if p is on the boundary of the convex hull of P. In Figure 2.2, V_{p_1} and V_{p_3} are unbounded and p_1 and p_3 are on the convex hull boundary. The Delaunay triangle $p_1p_2p_3$ is dual to the Voronoi vertex v and the Delaunay edge p_1p_2 is dual to the Voronoi edge uv. For Euclidean space R^2 and R^3 (2-dimensional and 3-dimensional space), Table 2.1 demonstrates all corresponding geometry elements in dual relationship between Voronoi diagram and Delaunay triangulation. Additionally, for the Delaunay triangle, $p_1p_2p_3$, consider a circumcircle, which is the unique circle passing through p_1, p_2 and p_3. Its center is the corresponding Voronoi vertex v and it encloses no other point in P. It turns out that empty circles characterize Delaunay triangles in Euclidean space R^2. Analogously, Delaunay tetrahedrons in Euclidean space R^3 are said to have the empty ball property.

Table 2.1 Corresponding geometry elements in dual relationship between Voronoi diagram and Delaunay triangulation

		Voronoi Cell (Voronoi Face)	**Voronoi Edge**	**Voronoi Vertex**
Euclidean Space R^2		Convex polygon	Line	Point
		Point	Line	Triangle
		Delaunay Vertex	**Delaunay Edge**	**Delaunay Cell (Delaunay Face)**
	Voronoi Cell	**Voronoi Face**	**Voronoi Edge**	**Voronoi Vertex**
Euclidean Space R^3	Convex polyhedron	Convex polygon	Line	Point
	Point	Line	Triangle	Tetrahedron
	Delaunay Vertex	**Delaunay Edge**	**Delaunay Face**	**Delaunay Cell**

Gabriel simplex

A simplex is called Gabriel if its smallest circumscribing ball is empty [3]. All Gabriel simplices are subset of the Delaunay triangulation. As a 2-dimensional example, Figure 2.3 shows a point set $P = \{p_1, p_2, p_3, p_4\}$, all simplices of its Delaunay triangulation $D(P)$ possess four vertices (0-simplices) p_1, p_2, p_3 and p_4; five edges (1-

simplices) p_1p_2, p_2p_3, p_3p_4, p_4p_1 and p_2p_4; two triangles (2-simplices) $\Delta p_1p_2p_4$ and $\Delta p_2p_3p_4$. Only the smallest circumscribing ball (dashed circle) of edge p_2p_4 is not empty (enclosing another vertex p_1 in P). Hence all edges are Gabriel edges except edge p_2p_4. Gabriel graph is well-known and extensively used geometric graph that only contains all these Gabriel edges (1-simplices), denoted as $GG(P)$. All Gabriel triangles (2-simplices) of point set P are denoted as $GT(P)$, where $GT(P) \subseteq DT(P)$. Gabriel simplex is often used in the initial step of surface reconstruction to help determine the final desired output.

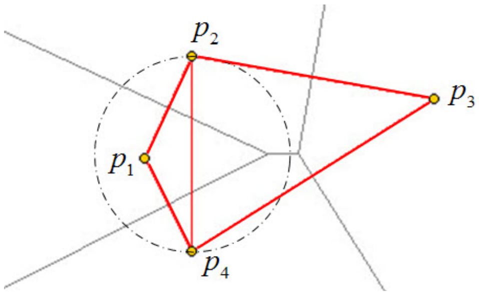


Figure 2.3 An example of Gabriel graph in the plane

2.1.2. Related Concepts

K-ball and *K*-sphere

In fact, Euclidean space R^k is a topological space, whose topology is the system of open sets. In this system, each open set is a union of open balls set, which is defined as the set of all points closer than certain distance from a given point. As described the book by Dey [4], let x denote a point in R^k, that is, $x = \{x_1, x_2, ..., x_k\}$ and

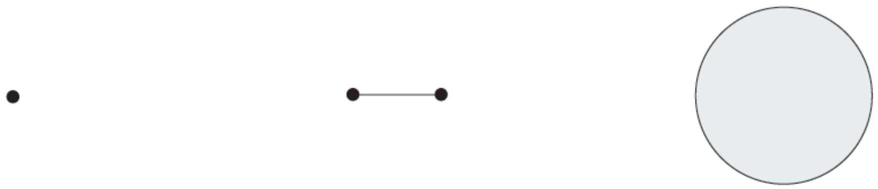
$\|x\| = (x_1^2 + x_2^2 + ... + x_k^2)^{1/2}$ denote its distance from the origin. Example of its subspace topology are the *k*-ball B^k, *k*-sphere S^k, and the open *k*-ball B_o^k where [4]

$$B^k = \{x \in R^k \mid \|x\| \le 1\}$$

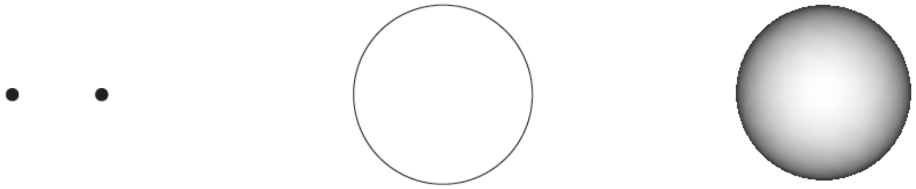
$$S^k = \{x \in R^{k+1} \mid \|x\| = 1\}$$

$$B_o^k = B^k \setminus S^k$$

Some examples of *k*-ball and *k*-sphere are shown in Figure 2.4.



0-ball (point) (a)
1-ball (closed interval) (b)
2-ball (closed disk) (c)



0-sphere (pair of points) (d)
1-sphere (circle) (e)
2-sphere (usual sphere) (f)

Figure 2.4 Some examples of *k*-ball and *k*-sphere in [5]

Homeomorphism

Two topological spaces are the same when one has a correspondence to the other which keeps the connectivity unchanged. For instance, the surface of a sphere can be deformed into a cube without any incision or attachment during the deformation process.

They have the same topology. A precise definition for this topological equality is given by a map called homeomorphism. A homeomorphism between two topological spaces is a map $f : T_1 \rightarrow T_2$ which is bijective, continuous and has a continuous inverse [4]. If a homeomorphism exists, T_1 and T_2 are homeomorphic. In practice, two homeomorphic topological spaces are often called topologically equivalent.

Two homeomorphic surfaces in Euclidean R^3 have the same properties and neighborhoods, which can be completely identified by their genus G, i.e., the number of through-holes. Figure 2.5 shows some topological spaces some of which are homeomorphic. Figure 2.5a is the 1-ball and is homeomorphic to both Figure 2.5b and Figure 2.5c spaces. Figure 2.5d is the 2-ball and homeomorphic to Figure 2.5e space. An open 2-ball in Figure 2.5f is not homeomorphic to the 2-ball in Figure 2.5d.

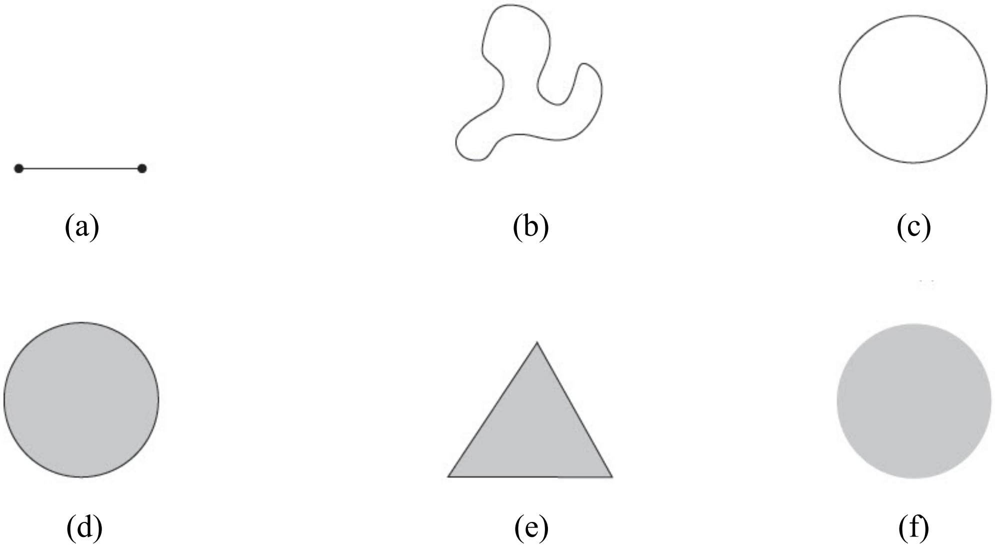

(a) (b) (c)
(d) (e) (f)

Figure 2.5 Examples of homeomorphism in [4]

Manifolds

Manifolds are particularly nice topological spaces defined locally [5]. A topological space is a k-manifold if each of its points has a neighborhood homeomorphic to the open k-ball which in turn is homeomorphic to R^k [4]. Here only k-manifolds that are subspaces of the Euclidean space are considered. For example, the plane, the sphere and torus with one through-hole all are 2-manifolds. The number of through-holes in a 2-manifold is called its genus G.

As discussed previously, an important topological quality of a surface is whether or not it is 2-manifold, which is the case if for each point the surface is locally homeomorphic to a disk (or a half-disk at boundaries) [1]. For triangle mesh surface, 2-manifold means that a triangle mesh does neither contain non-manifold edges or non-manifold vertices, nor self-intersections. Some non-manifold examples of triangle mesh are shown in Figure 2.6. A non-manifold edge (Figure 2.6b) has more than two connected triangles and a non-manifold vertex (Figure 2.6a) is generated by pinching two surface patches together at that vertex. The plot in Figure 2.6c is a non-manifold case. Non-manifolds in triangle meshes are often fixed by post-processing in mesh processing.

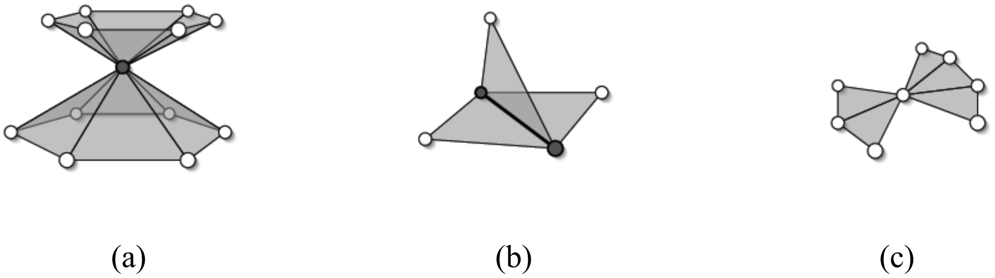

(a) (b) (c)

Figure 2.6 Non-manifold cases of triangle mesh in [1]

Both computational geometry and point set topology are well-established branches in mathematics. Some basic concepts and terminology briefly introduced here can help understand the setup of surface reconstruction algorithms in the following chapter. More details on point set topology and computational geometry can be found in these books [5-8]. A number of useful mathematical definition or concepts in surface reconstruction are collected in the book by Dey [4].

2.2. Existing Surface Reconstruction Approaches

As mentioned before, surface reconstruction refers to the conversion of a discrete point cloud representation into a continuous surface representation. If the discrete point cloud data has enough resolution to represent the scanned model surface geometry, the reconstructed surface would recover both the topology and geometric shapes successfully. In the past few decades, many surface reconstruction algorithms have been proposed for various applications, depending on properties of the input point cloud data, requirement of the output surface, user preference, and so on. These surface reconstruction algorithms are often classified into three main categories: implicit surface, region growing, and Delaunay-based approaches [9].

2.2.1. Implicit Surface

In the implicit surface approaches, the basic idea is to use the input point cloud to build a function in the Euclidean space R^3. The function is formulated to be negative inside and positive outside of the modeled object. The desired surface can then be extracted as the zero level set of the formulated function. In general, the implicit surface approach can output a watertight manifold mesh, which is required to approximate all points in point cloud data. Implicit surface approach is robust for noisy input point cloud data due to its approximation. Figure 2.7a and Figure 2.7b show a point cloud $P \subset R^2$ interpolated and approximated by a curve in the plane. The approximation can smooth the noise and result in a well-behaved surface. However, the goodness of fit can not be easily controlled by approximation and implicit surface approach maybe output some spurious components in reconstructed surface.

(a) (b)

Figure 2.7 Interpolation and approximation of a 2D point set in [10]

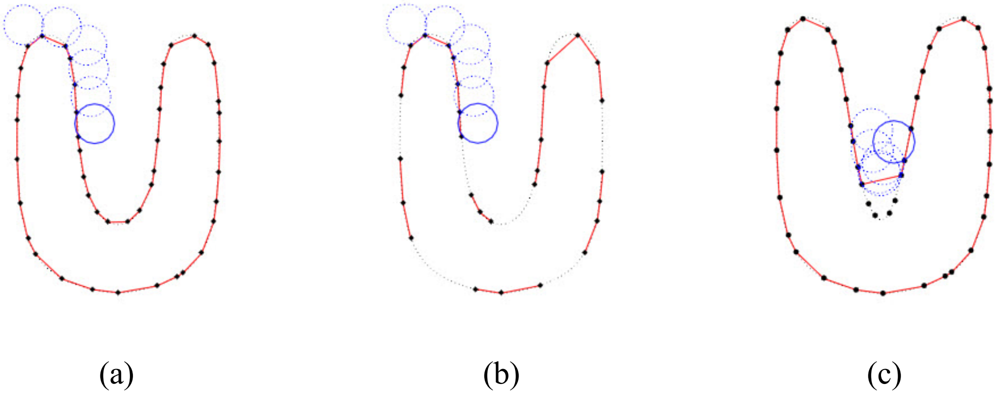
In 1992, Hoppe et al. [11] firstly proposed a implicit surface method for surface reconstruction, which estimates the normal vector for point cloud data by plane fitting technology in the initial stage. Then a distance function $f(x)$ is defined to negative inside of the object and positive outside of it. Finally, the zero-set of $f(x)$ is extracted as the desired surface and a piecewise linear triangle mesh is yielded through Marching Cubes algorithm. Based on the similar method, Curless and Levoy [12] focused on the problem of surface reconstruction from laser scanning data. They also built a signed distance function and get their desired output by an iso-surface extraction step. In addition, they consider some special issues on laser scanning range images integration.

Their algorithm can output a reconstructed surface with good quality and is computationally efficient.

The weighted sum of basis functions is often used in implicit surfaces, especially the radial basis functions. Implicit surfaces based on the radial basis functions have been applied in algorithms of Turk and O'Brien [13] and Dinh et al. [14] to reconstruct surface. Their methods address some issues in real data sets, such as noise, non-uniform distribution and sparsity. Carr et al. [15] used polyharmonic radius basis functions to reconstruct surface, whose output is a smooth and manifold mesh. Another method on implicit surface representation try to build many implicit functions locally adjacent to the point cloud, and then a function $f(x)$ is formulated by blending them together. The multilevel partition of unity (MPU) surface representation [16] is proposed based on this method. The space around the point cloud is decomposed by octree data structure. Each octree leaf contains a fixed number of points, which also includes an associated normal. The MPU algorithm is computationally efficient and can handle hundreds of thousands of input point cloud data. Except the concept of partitions of unity, another application based on this method is formulated in Moving Least Squares (MLS) function approximation. An approximating quadric polynomials f_i is computed to cover homogenous patches of the desired surface in the proposed algorithm of Xie et al. [17]. Those patches are extended from seeds and incrementally grown as long as a quadric can approximate the points included well. Other examples of algorithms in this category are listed in [18-23].

2.2.2. Region Growing

The region growing approach begins with a seed triangle and incrementally grows or expands from this seed triangle until the complete point cloud data set is covered. Unlike the implicit surface method, the region growing approach takes every point in the point cloud as the reconstructed triangle mesh vertex (interpolating all points in point cloud). Therefore, they will keep the details of the original surface of physical object and the reconstructed surface is expected to be more accurate.



(a) (b) (c)

Figure 2.8 A 2-dimensional example of BPA algorithm [24]

The reputable ball-pivoting algorithm (BPA) of Bernardini et al. [24] falls into this category. The basic procedure behind BPA algorithm is simple: starting with a seed triangle, a ball of user-specified radius *r* lying on this triangle (touching its three vertices). This ball is pivoted around an arbitrary edge of the current boundary, which is just the edges of the seed triangle, until it touches another sample point. If the ball touches another point in point cloud on its rotation movement around the edge, a new triangle can be built between the boundary edge and this point. New boundary is thus created and the rotation movement continues. As the ball rotates on the sample points the triangle meshes incrementally grow until all point cloud data are covered. If there exist separately connected mesh patches, another new seed triangle is chosen and this rotation process is repeated. BPA algorithm generates an interpolating triangle mesh from a given unorganized point cloud data. An oriented normal vector at each point is assumed to be available and that a global minimum threshold can be estimated for the density of point cloud. The outstanding issues with this approach are the identification of appropriate seed triangles and the determination of user-specified parameter. A 2-dimensional BPA example is shown in Figure 2.8. Figure 2.8a is a successful reconstructed curve in the plane by BPA algorithm. Figure 2.8b demonstrates the user-specified radius r is too small that some edges cannot be created with low point cloud density. Figure 2.8c demonstrates another failure case that the user-specified radius r is too large to reconstruct some high curvature regions.

Much effort has been invested in this approach recently. Huang and Menq [25] proposed a combinatorial growing process to build the 2-dimensional manifold triangle mesh directly without the need of an intermediate 3D representation. The output mesh was able to achieve second-order approximation to surface geometry of the original object. Their method is computationally efficient but with the same common drawback that the reconstruction quality heavily depends on the user-specified parameters, which are difficult to assign due to their close relationship with the point cloud density. Lin et al. [26] presented an improvement based on an intrinsic property of the point set, namely, the sampling uniformity degree. They tried to mitigate the limitation of the user-specified parameters. Li et al. [27] proposed a priority-driven region growing method which seeks to progressively construct the surface mesh from smooth to sharp regions. The shape deviation of at the boundary of the mesh growing area is considered in their method and a priority queue to the advancing front of the mesh area is built based on these shape deviations. The mesh growing process is then driven by the priority queue and the complex geometry or topology of the original surface of physical object can be successfully reconstructed.

Additionally, some researchers [9, 28, 29] modified the region growing method by picking triangles from Delaunay triangles to reconstruct the desired triangle mesh surface. Comparing with the traditional region growing approaches, their reconstructed surface appeared more systematic and robust because it inherits the structural characteristics of the Delaunay triangulation, which nicely complements the absence of geometric information in a set of unorganized point cloud data.

2.2.3. Delaunay-based

Both Delaunay triangulation and its dual Voronoi diagram are essential geometric data structures in computational geometry and provide a powerful way to approximate the neighborhoods in the discrete domain. They are able to explore the neighborhood of every point in a point set $P \subset R^3$ in all relevant directions. The Delaunay-based approach aims to extract a collection of triangles from the complete set of Delaunay triangles to construct the desired triangle mesh surface. A 2-dimensional curve reconstruction example based on Delaunay triangulation is illustrated in Figure 2.9. A point set $P \subset R^2$ is shown in Figure 2.9a and its Delaunay triangulation is shown in Figure 2.9b. Notice the Delaunay triangulation can capture neighbors in all directions, no matter how non-uniform the point set $P \subset R^2$ behaves. Figure 2.9c demonstrates

Delaunay-based algorithms attempt to identify or extract a correct subset from Delaunay triangulation for the desired curve through various filtering methods or geometric heuristics. The successfully reconstructed curve is drawn in Figure 2.9d.

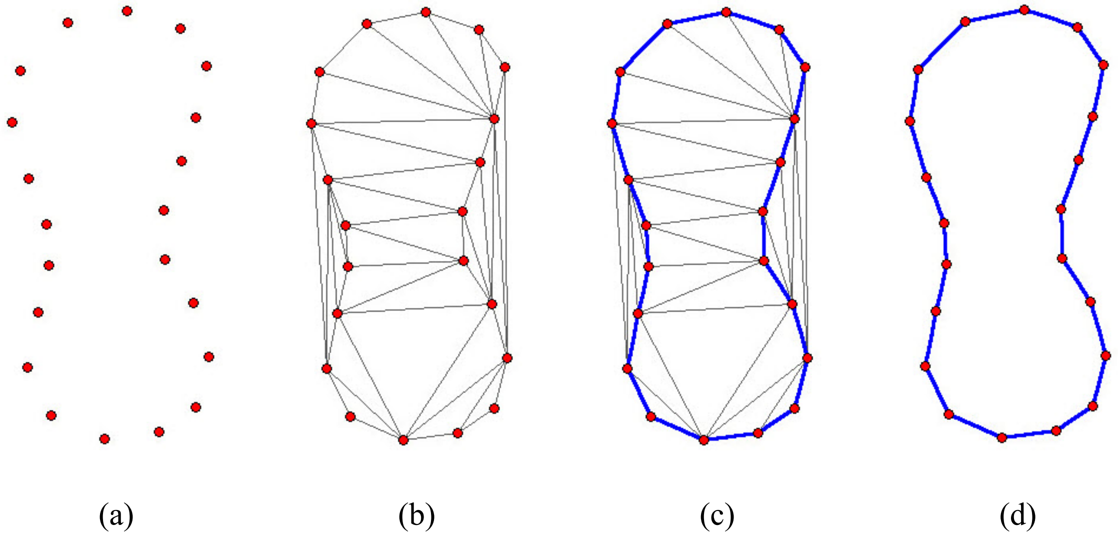

(a) (b) (c) (d)

Figure 2.9 A 2-dimensional example of Delaunay-based surface reconstruction

Boissonnat [30] appeared to be the first researcher to introduce such a Delaunay-based triangle mesh reconstruction algorithm. He tried to sculpt the shape from the 3-dimensional Delaunay triangulation. Those tetrahedrons likely to be outside the object are identified by their geometry shape and removed one by one, thus the remaining solid is always a sphere. Edelsbrunner and Mücke [31] used a filter of the Delaunay triangulation, the well-known alpha shape, which in face is a generalization of the convex hull and sub-graph of the Delaunay triangulation. Triangles with small circumspheres are retained as possible surface triangles. To date, more and more algorithms along this line have been proposed such as an improved alpha shape algorithm by Xu and Harada [32], the gamma-neighborhood graph by Veltkamp [33] and the umbrella filter algorithm by Adamy et al. [34].

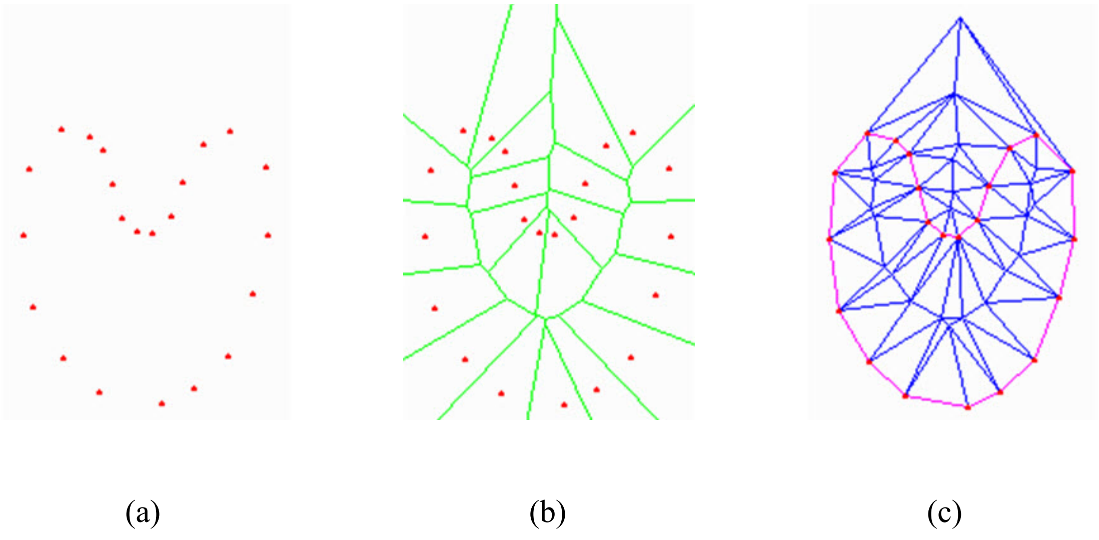

(a) (b) (c)

Figure 2.10 A 2-dimensional example of the crust algorithm [35]

Amenta and her co-workers [36, 37] presented the crust algorithm, the first algorithm with a theoretical guarantee for topological correctness of the generated mesh for points sampled from a smooth surface. The resulting mesh is homeomorphic and geometrically close to the original smooth and non-sharp object surface when the prescribed sampling condition is satisfied. Also they were the first to propose the definition of poles, which are a subset of the Voronoi vertices of the point set P and can represent the approximated medial axis when P is a sufficiently dense point cloud. Figure 2.10 illustrates the basic idea of a 2-dimensional crust reconstruction algorithm for a curve in the plane: Let $P \subset R^2$ be a finite point cloud in the plane (Figure 2.10a) and let V be the vertices of $V(P)$, the Voronoi diagram of P. Let P' be the union of P and V (all red points and intersection points between two green lines in Figure 2.10b). Let

$D(P')$ be the Delaunay triangulation of P'. An edge of $D(P')$ belongs to the crust of P if both its endpoints belong to P (red lines in Figure 2.10c).

Additionally, Amenta et al. [38, 39] proposed an extended Delaunay-based algorithm based on the crust algorithm: the well-known power crust algorithm. They used a weighted Voronoi diagram of the poles, known as a power diagram, rather than the Delaunay triangulation of the point set P. The power crust algorithm can generate a watertight mesh surface. Unfortunately, it also introduces many extra reference points in its output and does not produce a pure triangle mesh surface. The cocone algorithm by Amenta et al. [40] improved their previous method. They observed that the vector from each point p in point set P to either of its poles could be regarded as the approximation of the surface normal. Hence Delaunay triangles lying on the desired surface can be identified by comparing their normal vectors with the vectors to the poles. This algorithm was extended in different ways by Dey and others, to treat surfaces issues on sharp features and boundaries [41]. Additionally, it is also used to yield watertight triangle mesh surface [2]. A generalized definition of poles is proposed in the algorithm of Dey and Goswami [42] to include all Voronoi vertices far from the surface, which can be determined from noisy or smooth data. This work made their algorithm working well on the noisy inputs.

2.3. Outstanding Issues

As stated previously, many computer-aided design and inspection applications demand an accurately reconstructed surface corresponding to a watertight manifold triangle mesh passing through the unorganized data points (scanned from the ultra-precise 3D laser scanning system). Although many surface reconstruction approaches have been proposed in the past few decades, automatic reconstruction of watertight manifold interpolation triangle-mesh surfaces with correctly represented sharp features remains an open research issue in surface reconstruction research.

In the survey of surface reconstruction, we known that, instead of one single dominant method, multiple approaches are available with their own advantages and disadvantages and are chosen depending on specific applications. Implicit surface approaches have the advantage that the output surface always remains watertight and they work well on the noisy point cloud. Nevertheless, these approaches can lead to poorly shaped triangle meshes in some cases and the goodness of fit can not be easily controlled. More importantly, the generated mesh only approximates the input point cloud and does not in general interpolate (pass through) all the given points. This limits its use for applications such as computer-aided inspection and geometric modeling where it is mandatory to constrain the measured points on the reconstructed surface. For all the existing region growing methods, they are computationally efficient but there is one common drawback that the reconstruction quality heavily depends on the choice of the seed triangle and the user-specified parameters, which cannot be easily assigned due to their close relationship with the point density. Since the quality of reconstructed surface is not good enough and certain post-processing, such as hole-filling, are often needed in order to obtain a watertight manifold surface.

Delaunay triangulation and its dual Voronoi diagram are essential geometric data structures in computational geometry that provide a powerful way to approximate the neighborhoods in the discrete point cloud data. A main drawback of Delaunay-based approach is that it is computational expensive to build the initial global Delaunay triangulation. In Euclidean space R^3 , the computational complexity of Delaunay triangulation is $\mathbf{O}(n^2)$, where $n = |P|$ is the size of the point cloud data. Fortunately, the worst case hardly ever occurs in practice. In most cases, the computational complexity of 3D Delaunay triangulation is expected to be $\mathbf{O}(n)$ or $\mathbf{O}(n \log n)$ [43]. Additionally, most of Delaunay-based approaches cannot work well on very noisy point cloud data due to the requirement of interpolating all point cloud.

However, local neighbourhood information from Delaunay triangulation and its dual Voronoi diagram supplements the absence of geometric information in discrete point cloud data and it makes Delaunay-based approach more systematic and robust. Voronoi diagram and Delaunay triangulation explores the neighbourhood of each point in point cloud P in relevant directions in a way that even handle any non-uniform data. The time-consuming computation is no longer a major concern with the advancement in computer hardware and development of improved Delaunay triangulation algorithms by the computational geometry community. Robust and efficient methods to compute the Delaunay triangulation in Euclidean space R^3 have existed in CGAL algorithm library [43]. As previous discussed, a separate denoising pre-processing of point cloud is often enforced as one step prior to surface reconstruction, this makes Delaunay-based approach a better choice to reconstruct a watertight manifold triangle mesh surface that interpolates all measurement points in scanned unorganized point cloud P with low-noise.

3 WATERTIGHT SURFACE RECONSTRUCTION VIA PROGRESSIVE UMBRELLA FACET MATCHING

3.1. Introduction

Along with the increased applications of modern 3D scanning technologies, point cloud is emerging as a new data format for representing the surface geometry of a scanned object. As shown in Figure 1.1 in chapter 1, surface reconstruction has become the dominant and challenging task in the geometric processing of converting this discrete point representation to a final digital model in computer. If the point cloud has enough resolution to represent the original object surface geometry, the reconstructed triangle mesh surface would recover the correct topology and reliably approximate the geometry of the original object surface. Although many surface reconstruction algorithms have been proposed in the past, high-quality surface reconstruction remains a practical challenge. Especially for computer-aided design and inspection applications, automatic reconstruction of watertight manifold triangle mesh surface that interpolates (passes through) all measurement points in scanned point cloud data remains an open research issue.

As previous discussed in chapter 2.1.1, Delaunay triangulation and its dual Voronoi diagram are essential geometric data structures in computational geometry and they are capable of laying out the neighborhood of every point in a point set in all relevant directions. Local neighbourhood information from Delaunay triangulation and its dual Voronoi diagram supplements the absence of geometric information in discrete point cloud data. Therefore, Delaunay-based approach is robust and more systematic in nature. It becomes a better choice to reconstruct a watertight manifold triangle mesh surface interpolating all measurement points in scanned unorganized point cloud P with low-noise.

As stated in chapter 2.2.3, there are many Delaunay-based algorithms for surface reconstruction proposed in the past few decades. However, generation of a watertight manifold surface with correct topology has remained a challenge for most existing Delaunay-based algorithms. Additional post-processing procedure is often required to produce a watertight manifold triangle mesh surface, such as the hole-processing or manifoldness processing. Adamy et al. [34] introduced an umbrella filter algorithm coupled with a linear-programming based topological post-processing module designed for topologically correct watertight triangle mesh reconstruction. In their method, numerical difficulty could still arise from the non-smooth or under-sampled surface region. Dey and Goswami [2] also attempted to reconstruct a watertight manifold triangle mesh interpolating all point cloud and proposed an extended algorithm based on the cocone algorithm of Amenta et al. [40], namely, the tight cocone algorithm. However, their approach remains in effect a post-processing algorithm to the cocone algorithm and encounters difficulty in computing a watertight manifold surface when the condition of locality of undersampling is not satisfied.

In this chapter, a new Delaunay-based algorithm is presented, which is driven by umbrella facet matching (UFM). This algorithm seeks to generate, in parallel, a fully matched, local 2-dimensional manifold triangle mesh at each point (resembling the shape of an open umbrella) from its Delaunay triangle set. An umbrella is regarded as a fully matched umbrella when it fully overlaps with its neighboring umbrellas. Different from the method of Adamy et al. [34], this fully matched umbrella at each point guarantees the generation of a watertight manifold triangle mesh without the need for additional hole-filling post-processing. The reminder of this chapter is organized as follows: in the following section, relevant concepts and terminology to the proposed UFM algorithm are introduced; details of the UFM algorithm is outlined in Section 3.3; Section 3.4 provides the implementation results; and concluding remarks are given in Section 3.5.

3.2. Relevant Concepts

3.2.1. Definition of the Umbrella

In the computational geometry community, the common definition of a surface is that of an orientable continuous 2-dimensional manifold embedded in the Euclidean space R^3. Intuitively, it can be described as the closed (watertight) boundary surface of a non-degenerative 3-dimensional solid. The non-degeneration means that the solid does not have any feature of zero thickness and the closed boundary surface is able to unambiguously separate the interior and exterior of the solid. An open surface with finite size is one that can be extended into a closed boundary surface by filling its hole(s). As stated previously, a triangle mesh surface reconstructed from a point set in R^3 is in effect a piecewise linear surface representation. Hence, the discussion on surface reconstruction in the following sections refers to the subject of closed (watertight) manifold triangle mesh surface reconstruction.

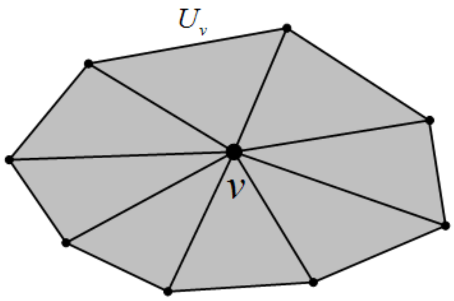
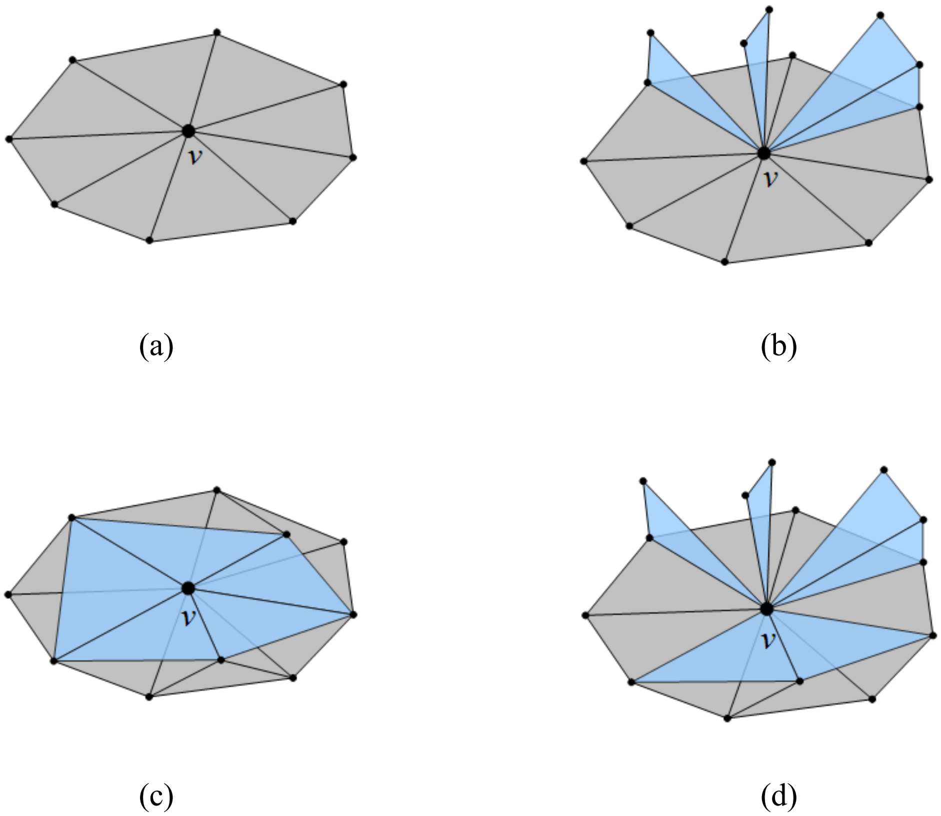


Figure 3.1 Umbrella U_v at a point v

As depicted in Figure 3.1, a local 2-dimensional manifold triangle mesh U_v at a point v is homeomorphic to a full disc. U_v is referred to as an (open) umbrella, which contains neither non-manifold edges/vertices nor self-intersections. Each point of a reconstructed watertight manifold triangle mesh should hold such an umbrella. Outgoing edges of the umbrella at v are all manifold edges linked back with point v and constitute the frame of the umbrella. The remaining edges of the umbrella are named as circumjacent edges, which connect the outgoing edges and form the profile of the umbrella. It is evident that, for each triangular facet of the umbrella U_v at v, it always consists of two outgoing edges and one circumjacent edge.

3.2.2. Delaunay Triangle Clusters

In constructing an umbrella at v from its Delaunay triangles in the present work, there exist four basic topological types of Delaunay triangle cluster incident to v, as shown in Figure 3.2. A fin is a triangular facet for which at least one of its two outgoing edges does not connect with any other facet (fins often appear in the redundant facet removal process when constructing an umbrella in this work). A pocket refers to the closed non-manifold triangular facet, where an outgoing edge is connected with more than two triangular facets. In fact, the complete set of Delaunay triangles incident to *v* corresponds to the type of umbrella with pockets (Figure 3.2c). Figure 3.2b and Figure 3.2d depict another two basic topological types: umbrella with fins and umbrella with fins and pockets. It should be noted, however, that an arbitrary subset of the complete Delaunay triangle set at *v* is likely not to correspond to any of these four basic topological types. Detailed description on the umbrella building process is to be presented in Section 3.3.1.



(a) (b)

(c) (d)

Figure 3.2 Topological types of Delaunay triangle clusters: (a) umbrella; (b) umbrella with fins; (c) umbrella with pockets; and (d) umbrella with fins and pockets.

3.3. Umbrella Facet Matching

In this work, the original object surface is a closed (watertight) surface. The input point cloud is a low-noise, unorganized coordinate data set containing no other geometric information (such as surface normals). There is no limitation on the genus of the original object. The basic objective of the UFM algorithm is to seek a fully matched umbrella at each point from its Delaunay triangles. The primary steps of this algorithm are as follows (more details will be presented in the subsequent subsections):

Notations: Point set $P \subset R^3$ with each point $v \in P$

$D(P)$**:** Delaunay triangle set of the complete point set P

DT_v: Delaunay triangle set incident to v

U_v**:** an umbrella at v

M_f : absolute matching index (evaluated from the matching results)

$M_{f(v)}$: relative matching index (evaluated from the matching results)

Step 1: Compute the Delaunay triangulation $D(P)$

Step 2: Establish an initial U_v for (each) v in parallel

Step 3: Update U_v in parallel according to the umbrella facet matching results

- **3.1:** Evaluate M_f and $M_{f(v)}$ for every facet of U_v;
- **3.2:** $U_v = DT_v$ (U_v starts/resets as the complete Delaunay triangle set at v);
- **3.3:** Generate a priority queue for all of the $U_v = DT_v$ facets according to the proposed priority queuing mechanism;
- **3.4:** Remove redundant facets (non-manifold facets in a pocket and fins) in U_v following the priority queue until U_v becomes a single umbrella for v;
- **3.5:** Repeat Steps **3.1** to **3.4** until U_v becomes a fully matched umbrella (the evaluated M_f value for every facet of U_v equals 3).

It should be pointed out that only one Delaunay triangulation computation for the complete point cloud data set is required. Once an initial umbrella at each point is constructed, overlap among these umbrellas can be evaluated and values of the parameter M_f and $M_{f(v)}$ for every umbrella facet can be determined based on the matching results. The algorithm then repeats Steps 3.1 to 3.4 to establish an updated umbrella from the complete Delaunay triangle set at each point. The umbrella at each point is constructed by removing redundant triangular facets in sequence according to the priority queue derived from the current matching results. This iterative process continues until all the fully matched umbrellas are found, which leads to a watertight manifold triangle mesh.

3.3.1. Building an Umbrella

Building an umbrella in this work is essentially a process that sequentially removes all redundant (non-manifold) triangular facets according to a priority queue. As described in Section 3.2.2, there are four basic topological types of Delaunay triangle cluster incident to a point v during the umbrella building process. A redundant facet is either a non-manifold facet in a pocket or a fin. The facet removal process starts with removing the non-manifold facet in a pocket, followed by an overall fin cleaning procedure. This means that whenever a fin exists, it is removed right away. This redundant facet removal process ends when there are no more non-manifold edges, vertices, or self-intersections in the updated facet cluster. The remaining triangular facets then correctly constitute an umbrella.

It is evident that different priority queues would lead to different umbrellas. It implies that a specific priority queue of the Delaunay triangles for v has to be found for building a desired umbrella at v. Since the desired umbrellas are fully matched ones in this work, the corresponding priority queue could be obtained by updating the priority queue based on the evaluation of the matching results of all the umbrella facets. For an existing umbrella, the matching results of its triangular facets can be evaluated and then used to establish an updated priority queue via a priority queuing mechanism. An updated priority queue then leads to an updated umbrella. This process repeats itself until the matching results show that a fully matched umbrella has been found. Therefore, the modules of the priority queuing mechanism, which generates the initial queue and evaluates the umbrella facet matching results, are clearly the core of the UFM algorithm.

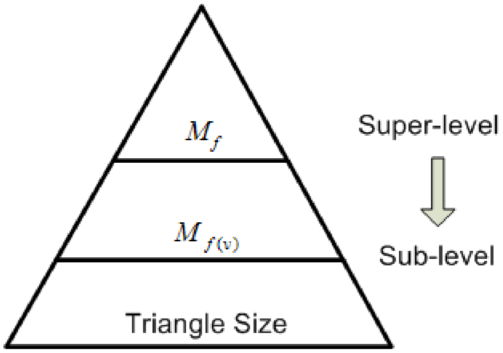


Figure 3.3 Priority queuing mechanism

3.3.2. Priority Queuing Mechanism and the Initial Queue

In order to construct the desired fully matched umbrella at every point, a priority queuing mechanism with three-level inheritance is introduced, where a sub-level always inherits the queuing from a super-level. This means that primary queuing rules should be placed in an ordered sequence from the most superior level downwards. For the priority queuing mechanism of this work (Figure 3.3), the queuing rule at the first (top) level is the absolute matching index M_f, representing the basic matching result; at the second (middle) level there is the relative matching index $M_{f(v)}$, representing the refined matching result; and the third (bottom) level is the size of the Delaunay triangle. Details of the matching indices M_f and $M_{f(v)}$ will be presented in the next section. In the initialization stage of building the initial/first umbrella, only the triangle size information is available (as no matching results exist yet). The diameter of the minimum circumsphere of the triangle is taken to quantify the triangle size. Therefore, the initial priority queue of $U_v = DT_v$ is established according to the minimum circumsphere diameters of the associated Delaunay triangles.

In this work, in order to build the initial umbrella more effectively, the subset of Gabriel facets of DT_v has been employed. A triangular facet in DT_v is a Gabriel facet if its minimum circumscribed sphere is empty of any point in the point set P. Evidently, all of the Gabriel facets are contained in DT_v and the Gabriel set GT_v incident to v is a subset of the Delaunay triangle set DT_v. The Gabriel set GT_v in general represents the geometry of the original object surface well [34, 44-46] and is thus taken as the first set of triangles to construct the initial umbrella. Nonetheless, as a subset of DT_v, the Gabriel set GT_v could in fact be a facet cluster that does not contain an umbrella at all. To ensure that an umbrella is established at every point in the initialization stage, the algorithm will seek an umbrella from the complete Delaunay triangle set DT_v once it is deemed necessary. The following is the detailed breakdown of Step 2 to establish an initial umbrella U_v for (each) point v in our algorithm:

2.1: $U_v = GT_v$ (U_v starts as the Gabriel set at v);

2.2: Generate a priority queue for all the facets in U_v according to their minimum circumsphere diameter (large to small);

2.3: Remove all the redundant non-manifold facets (fin or pocket triangles) in U_v following the priority queue;

2.4: Assert $U_v == An\,Umbrella$; otherwise, let $U_v = DT_v$ and go to Step **2.2**.

It can be seen from the above that the UFM algorithm first attempts to establish an initial umbrella for each point from its Gabriel set by removing redundant non-manifold triangles according to their minimum circumsphere size. If an umbrella cannot be found, the UFM algorithm then resort to the complete Delaunay triangle set to ensure that an initial umbrella is established at each point.

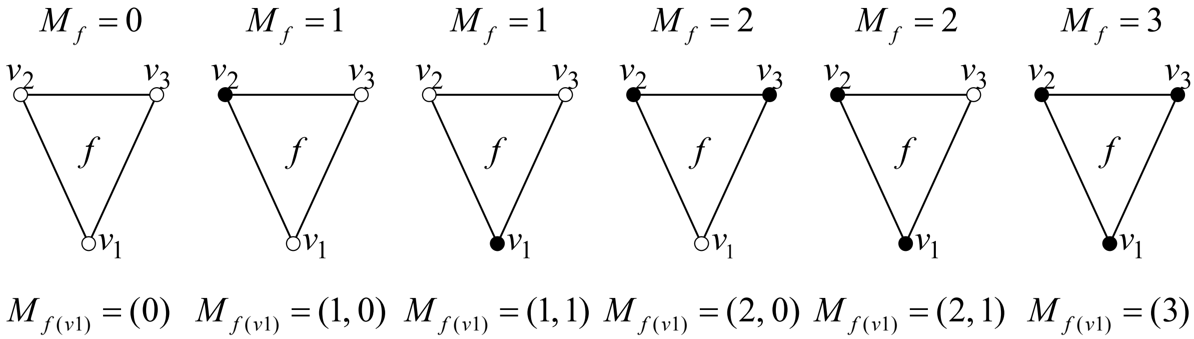


Figure 3.4 Absolute and relative matching indices of f at v

3.3.3. Evaluation of the Matching Results

After the initial umbrella at each point is established, two matching indices are devised to indicate the degree of overlap among the established umbrellas (Step 3.1). In other words, these matching indices are introduced to evaluate how much an umbrella overlaps with its neighboring umbrellas. In this work, the facet matching results are considered at two levels: basic and refined. The basic matching result is quantified by the absolute matching index M_f and the refined matching result is quantified by the relative matching index $M_{f(v)}$. The absolute matching index M_f is devised to indicate the degree of matching for a facet f. The relative matching index $M_{f(v)}$ is devised to indicate the degree of matching for f relative to the vertex v.

There are a total of six possible cases of M_f and $M_{f(v)}$ as shown in Figure 3.4. In essence, $M_{f(v)}$ is an extension of M_f. In the figure, the facet f has three vertices: v_1, v_2, and v_3. When M_f equals 3, this means that all three umbrellas incident to v_1, v_2, and v_3 include the facet f. The last (right most) figure in Figure 3.4 depicts this case. A solid dot for a vertex indicates that the umbrella of this vertex includes the facet f and an empty dot for a vertex indicates that its umbrella does not include the facet f. When M_f equals 2, only two of the three umbrellas that are incident to v_1, v_2, and v_3 include the facet f. In this case, there exist two possible situations. With respect to the vertex v_1, one situation is that the umbrella at v_1 does not include the facet f (the fourth figure in Figure 3.4) and the other situation is that the umbrella includes the facet f (the fifth figure in Figure 3.4). Their relative matching indices are then expressed as: $M_{f(v1)} = (2,0)$ and $M_{f(v1)} = (2,1)$ respectively. The first figure in Figure 3.4 illustrates the situation that all three umbrellas incident to v_1, v_2, and v_3 do not include the facet f. As a result, $M_f = 0$ and $M_{f(v1)} = (0)$. All possible matching index values for the facet

f are summarized in Table 3.1, where a check mark for an umbrella indicates that the umbrella includes the facet f.

Table 3.1 List of possible matching index values for facet $f(v_1, v_2, v_3)$

M_f	$M_{f(v1)}$	$M_{f(v2)}$	$M_{f(v3)}$	U_{v1}	U_{v2}	U_{v3}
3	(3)	(3)	(3)	√	√	√
2	(2, 1)	(2, 1)	(2, 0)	√	√	
2	(2, 0)	(2, 1)	(2, 1)		√	√
2	(2, 1)	(2, 0)	(2, 1)	√		√
1	(1, 1)	(1, 0)	(1, 0)	√		
1	(1, 0)	(1, 1)	(1, 0)		√	
1	(1, 0)	(1, 0)	(1, 1)			√
0	(0)	(0)	(0)			

Figure 3.5 illustrates typical matching results for an umbrella facet f with its vertex points v_1, v_2, and v_3. U_{v1}, U_{v2}, and U_{v3} represent the established umbrellas at v_1, v_2, and v_3 respectively. Both U_{v1} and U_{v2} include the facet $f(v1, v2, v3)$ and U_{v3} does not. In this situation, the absolute matching index for f is 2 ($M_f = 2$). For v_1 and v_2, the relative matching indices $M_{f(v1)} = (2,1)$ and $M_{f(v2)} = (2,1)$. For v_3, the relative matching index $M_{f(v3)} = (2,0)$. If the umbrella U_{v3} incident to v_3 is updated to U'_{v3} (still incident to v_3) which now includes the facet f, $M_f = 3$ and $M_{f(v1)} = M_{f(v2)} = M_{f(v3)} = (3)$ can be achieved. The facet $f(v1, v2, v3)$ is then called a matched facet. An umbrella for which all of its triangular facets are matched facets is a fully matched umbrella. When the fully matched umbrella at every point of the point

cloud data set is found, all the matched umbrellas constitute a watertight manifold triangle mesh.

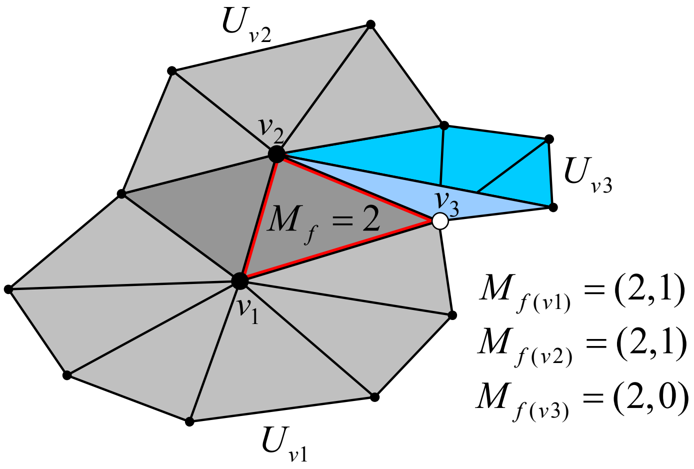


Figure 3.5 Typical matching results for an umbrella facet

As stated before, the proposed UFM algorithm centers on the sequential removal of redundant triangular facets from the candidate facet cluster. This is achieved via the priority queuing mechanism with three-level inheritance based on the umbrella facet matching results. The priority queue is formed according to the evaluated value of the relative matching index $M_{f(v)}$, which is devised to inherit the evaluated value of the absolute matching index M_f. For all the triangular facets from the candidate facet cluster at v, the sequence is formed from $M_{f(v)} = (0)$, $M_{f(v)} = (1,0)$, $M_{f(v)} = (1,1)$, $M_{f(v)} = (2,0)$, $M_{f(v)} = (2,1)$, to $M_{f(v)} = (3)$. Those facets with the same $M_{f(v)}$ value are then ordered by their minimum circumsphere radii, as the bottom-level rule in the priority queuing mechanism. Evidently, the priority queue will be continually updated at each point until all the fully matched umbrellas are successfully found.

In the implementation of the proposed UFM algorithm, a special scheme has been introduced for which a temporary constraint regarding facet removal can be inserted. For example, in order to speed up the convergence of the involved numerical iterations, triangular facets with $M_{f(v)} = (3)$ can be set as non-removable as they are in general good facets. In fact, facets with any matching index values can be set as non-removable through this temporary constraint scheme to achieve a desired property in the reconstructed mesh.

3.3.4. Computational Complexity

The flowchart of proposed Umbrella Facet Matching algorithm is shown in Figure 3.6. In the worst case, the computational complexity of 3D Delaunay triangulation in Step 1 of the UFM algorithm is $\mathbf{O}(n^2)$, where n is the number of points in the point cloud data set. Fortunately, the worst case hardly ever occurs in practice. In most cases, the computational complexity of 3D Delaunay triangulation is expected to be $\mathbf{O}(n)$ or $\mathbf{O}(n \log n)$ [43]. Let m be the number of Delaunay triangles from the 3D Delaunay triangulation of the point cloud. The computational complexity to establish an initial umbrella at every point (the umbrella initialization process of Step 2) is $\mathbf{O}(m \log m)$ because the Delaunay triangles incident to every point have to be sorted into a priority queue according to their circumsphere radii. For updating the umbrellas according to the umbrella facet matching results (Step 3), the computational complexity is still $\mathbf{O}(m \log m)$ due to the queuing of the involved triangular facets.

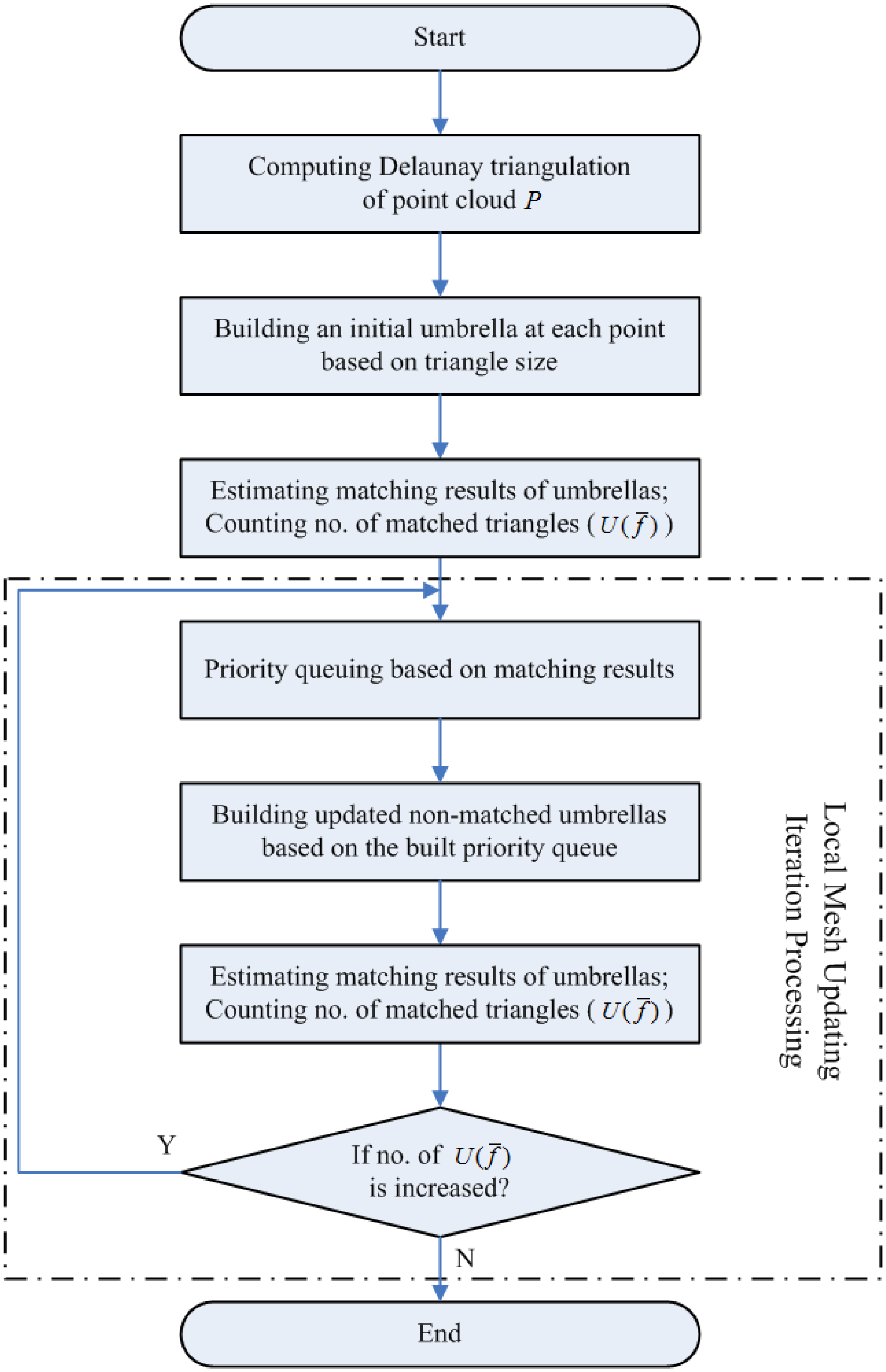

Start
Computing Delaunay triangulation of point cloud P
Building an initial umbrella at each point based on triangle size
Estimating matching results of umbrellas; Counting no. of matched triangles ($U(\bar{f})$)
Priority queuing based on matching results
Building updated non-matched umbrellas based on the built priority queue
Estimating matching results of umbrellas; Counting no. of matched triangles ($U(\bar{f})$)
If no. of $U(\bar{f})$ is increased?
Y
N
End
Local Mesh Updating Iteration Processing

Figure 3.6 Umbrella Facet Matching algorithm flowchart

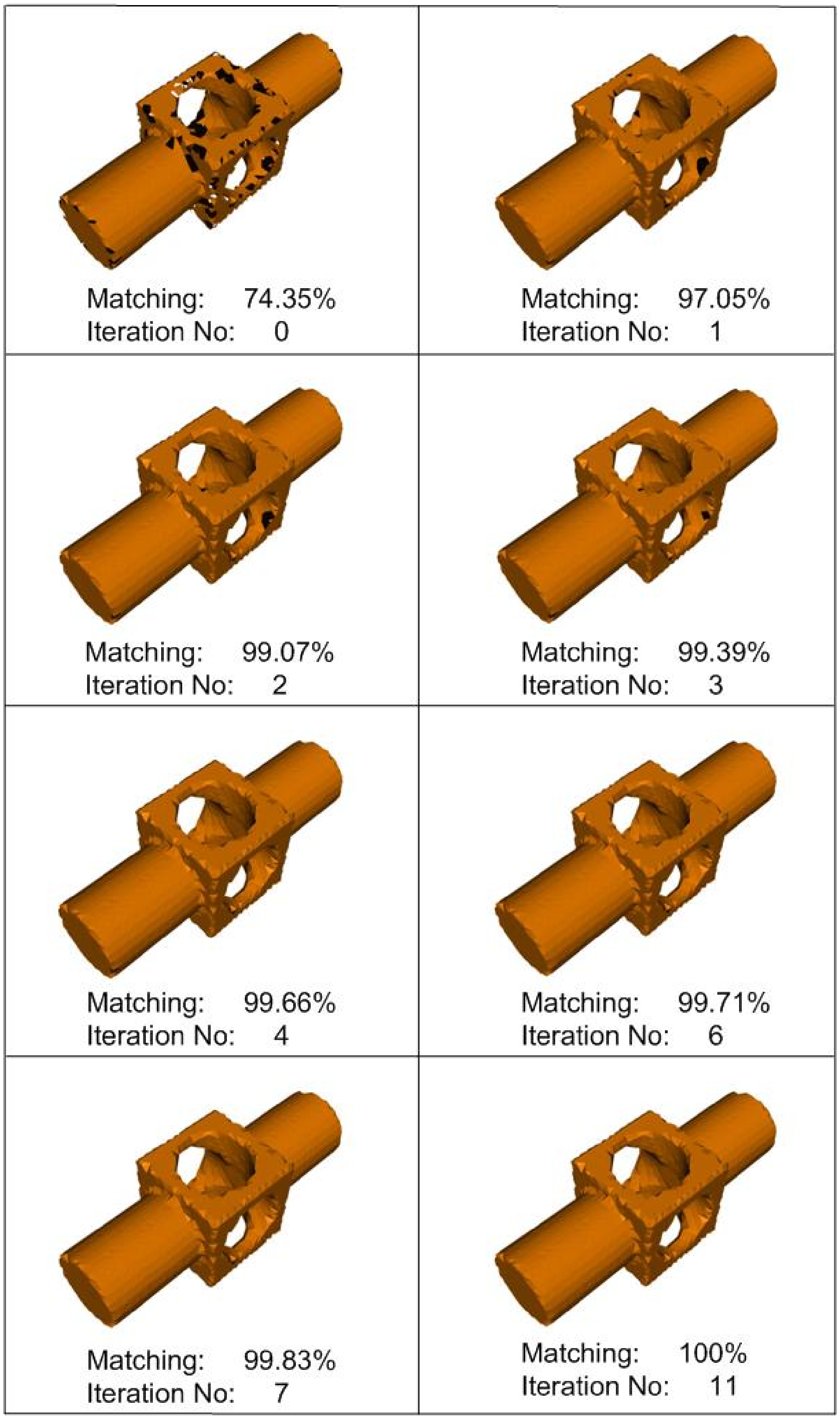


Figure 3.7 Detail UFM convergence process for the Mechpart data set

3.4. Implementation Results and Discussion

The proposed UFM algorithm has been implemented and evaluated using many known point cloud data sets. To perform the 3D Delaunay triangulation (DT) of a point set, existing codes in the Computational Geometry Algorithms Library CGAL [43] were employed. Also, because of the need to maintain topological information, another open-source template library, the VCG Library [47], was referenced for manipulating and processing the triangle meshes. The associated case studies were carried out on a Windows-based PC with a 2.66GHz processor and 4GB memory.

Table 3.2 lists the computed results for the test point cloud data sets downloaded from the Internet. It can be seen that with the exception of the Casting Die data set, 100% matching has been attained for all the test cases of different genus. The matching percentage represents the ratio of the number of the resulting points with fully matched umbrellas to the number of points in the input point cloud. A matching ratio of 100% means that fully matched umbrellas at all the points have been found and a watertight manifold triangle mesh is successfully reconstructed. As a typical case, Figure 3.7 shows the detailed convergence process for the Mechpart data set. The umbrella matching was at 74.35% when the initial umbrellas for the point set were established. After 11 iterations, the fully matched umbrellas for all the points were found and the total computational time was only 7.83 seconds for our moderate PC computing platform. It should be emphasized here again that no hole-filling post-processing was needed in our algorithm. Once the algorithm converges, the reconstructed mesh will be guaranteed to be a watertight manifold triangle mesh. Also, it has been observed that the reconstructed meshes are consistently being homeomorphic and geometrically close to their respective original object surfaces. The reconstructed meshes do not need to include any extra points and they interpolate all the input points.

Table 3.2 Implementation results for some public point cloud data sets

Data Set			**Implementation Results**					
Name	Genus	Input (points)	Output (triangles)	Matching	ΔF	Computing Time (sec.)		
						DT	UFM	Total
Bunny	0	35,947	71,890	100%	0	8.39	14.48	22.87
Golf Club	0	16,585	33,166	100%	0	4.20	8.10	12.30
Mechpart	3	4,102	8,212	100%	0	1.48	6.35	7.83
3Holes	3	4,000	8,008	100%	0	1.48	1.86	3.34
Knot	1	10,000	20,000	100%	0	3.54	7.20	10.74
Mannequin	0	12,772	25,540	100%	0	3.33	8.53	11.86
Casting Die	0	63,613	127,003	99.7%	219	14.95	108.63	123.58
Oilpmp*	0	30,937	61,862	100%	0	7.42	34.36	41.78
Rocker Arm	1	10,044	20,088	100%	0	2.25	7.48	9.73
Screwdriver	0	27,152	54,300	100%	0	6.84	14.27	21.11
Hand	0	25,001	49,998	100%	0	6.50	15.61	22.11
Teapot	1	25,667	51,334	100%	0	5.92	93.99	99.91

* *Oilpmp included 4 repeated points and they were removed.*

It is well-known in practice that the non-uniform point distribution in a point cloud often has strong impact on the quality of the associated reconstructed triangle surface mesh. As shown in Figure 3.8, the point distribution in the Mannequin data set is highly non-uniform with much higher point density in and around the ears, the mouth, and the eyes. This data set in fact represents an open surface with a relatively large hole at the bottom. It was, however, still treated as a watertight closed surface for the algorithm to try to fill this hole. The UFM algorithm successfully addressed these challenges and reconstructed a topologically correct watertight manifold triangle mesh for the Mannequin data set.

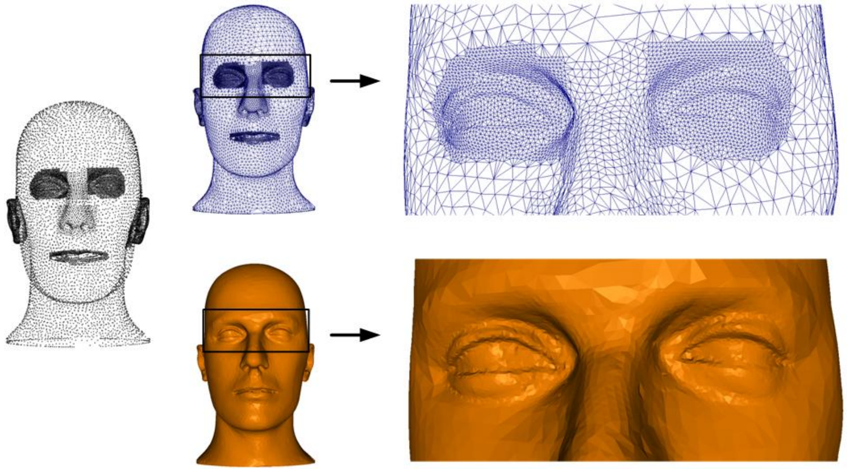


Figure 3.8 Reconstructed mesh for the Mannequin data set

As stated previously, the reconstructed topologically correct surface mesh will be a watertight manifold triangle mesh which is homeomorphic to the original object surface. This requires that the number of vertices (input points) and the resulting number of triangles in the reconstructed mesh must satisfy the following Equation 3.1 derived from the Euler's formula [34]:

$$F = 2 \cdot V + 4 \cdot (G - 1) \tag{3.1}$$

where F denotes the number of the reconstructed triangles, V the number of vertices, and G the genus of the original object. It is known that the quality of a reconstructed triangle mesh depends on the point density and distribution as well as the geometric feature complexity in the original object surface. It is very challenging for any Delaunay-based algorithm to guarantee topologically-correct surface reconstruction because of noise, inadequate input point density, and non-smoothness of the underlying surface feature. Like most existing algorithms, the UFM algorithm cannot theoretically guarantee that the reconstructed triangle mesh will always be homeomorphic to the original object surface. Nonetheless, the converged implementation results shown in Table 3.2 do indicate that all the reconstructed watertight manifold triangle meshes by the UFM algorithm are homeomorphic to the original object surface. The quality of a reconstructed triangle mesh was evaluated and quantified via Equation 3.2 as:

$$\Delta F = \left| 2 \cdot V + 4 \cdot (G-1) - F \right| \qquad \textbf{(3.2)}$$

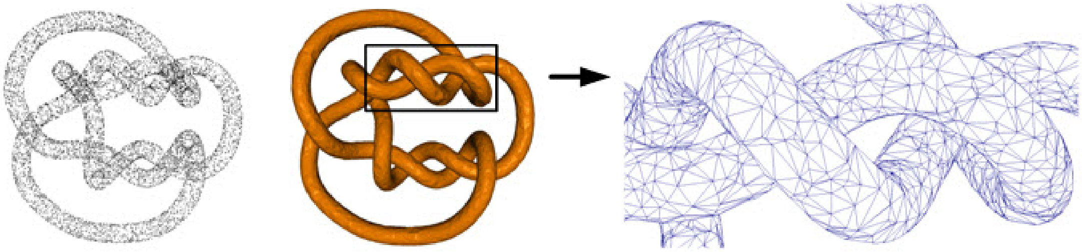


Figure 3.9 Reconstructed mesh for the Knot data set

Figure 3.9 depicts another representative case in Table 3.2. As shown in the left figure, the Knot data set is a non-uniformly sampled data set with a genus value of 1. The two figures on the right illustrate the well-reconstructed watertight manifold triangle mesh by the UFM algorithm. The reconstructed surface is homeomorphic to the original

Knot surface as ΔF equals 0. With the exception of the Casting Die data set, all the other computed results confirm that the reconstructed watertight manifold triangle meshes (with the matching values being 100%) are homeomorphic to the original object surfaces ($\Delta F's$ being 0). The resulting rendered images of the reconstructed triangle meshes for these data sets are shown in Figure 3.10.

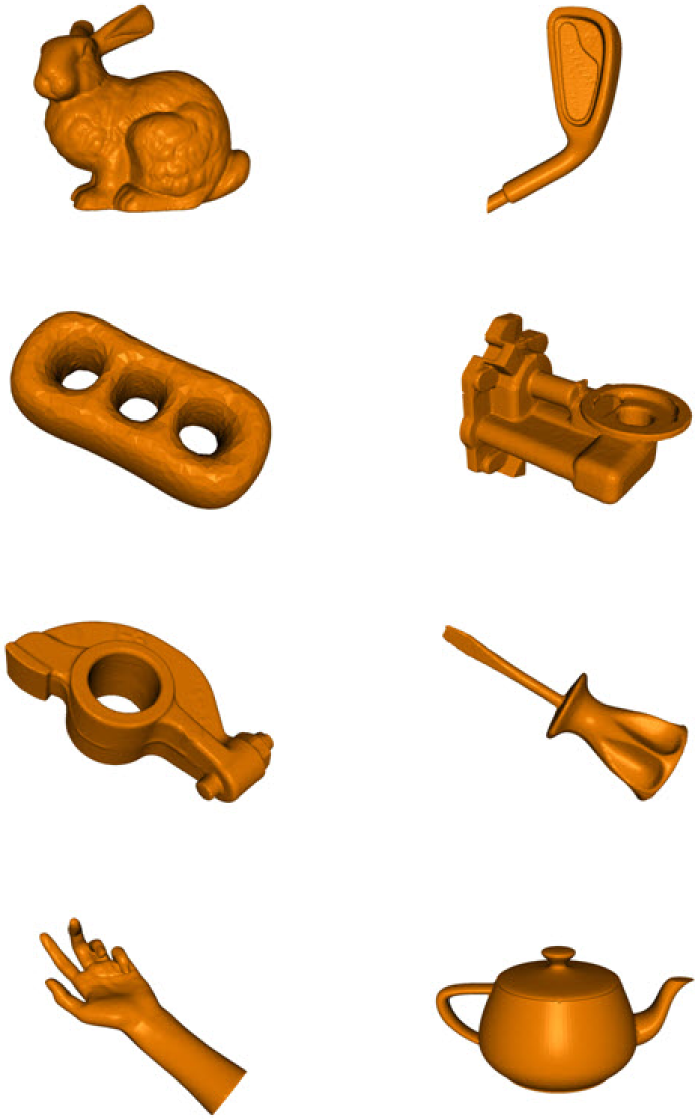


Figure 3.10 Other reconstructed meshes for the data sets in Table 3.2

Table 3.3 Comparison of the reconstructed meshes-algorithms not employing hole-filling post-processing

Data Set			Results		
Name	Genus	Input (points)	Algorithm	Output (triangles)	ΔF
Bunny	0	35,947	UFM	71,890	0
			[27]	71,669	221
Golf Club	0	16,585	UFM	33,166	0
			[34]*	33,308	142
3Holes	3	4,000	UFM	8,008	0
			[34]*	8,108	100
Knot	1	10,000	UFM	20,000	0
			[27]	19,317	683
			[32]	20,726	726
			[34]*	20,396	396
Mannequin	0	12,772	UFM	25,540	0
			[27]	24,405	1135
			[32]	29,537	3
			[34]*	25,646	106
Oilpmp	0	30,937	UFM	61,862	0
			[27]	61,617	253
			[34]*	62,873	1003
Rocker Arm	1	10,044	UFM	20,088	0
			[26]	20,092	4
Screwdriver	0	27,152	UFM	54,300	0
			[26]	54,321	21

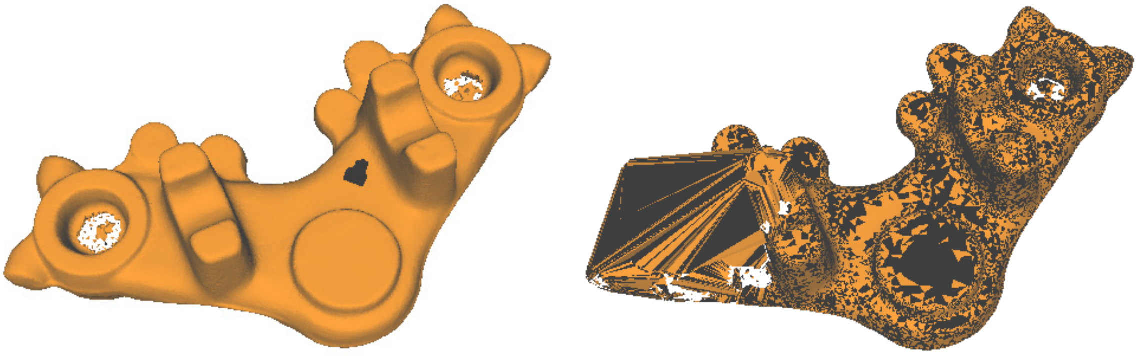
[*] Only the outputs without hole-filling post-processing in Ref. [34] are listed here.

Table 3.4 Comparison of the reconstructed meshes-algorithms employing hole-filling post-processing

Data Set			UFM		Tight Cocone [2]		Umbrella Filter [34]	
Name	Genus	Input (points)	Output (triangles)	ΔF	Output (triangles)	ΔF	Output (triangles)	ΔF
Bunny	0	35,947	71,890	0	71,886	4	-	-
Golf Club	0	16,585	33,166	0	33,158	8	33,166	0
Mechpart	3	4,102	8,212	0	8,212	0	-	-
3Holes	3	4,000	8,008	0	8,008	0	8,008	0
Knot	1	10,000	20,000	0	20,000	0	20,000	0
Mannequin	0	12,772	25,540	0	25,526	14	25,540	0
Casting Die	0	63,613	127,003	219	120,102	6901	-	-
Oilpmp	0	30,937	61,862	0	61,856	6	61,870	8
Rocker Arm	1	10,044	20,088	0	20,088	0	-	-
Screwdriver	0	27,152	54,300	0	54,280	20	-	-
Hand	0	25,001	49,998	0	49,998	0	-	-
Teapot	1	25,667	51,334	0	51,350	16	-	-

Table 3.3 lists the comparison of the reconstructed triangle meshes by different algorithms, without resorting to the post-processing step of filling possible holes, for the same publicly available data sets. According to Equation 3.2, lower ΔF values would indicate smaller topological difference and zero ΔF represents homeomorphism between the reconstructed mesh and the original object surface. In the computational tests, the reconstructed watertight triangle meshes by the proposed UFM algorithm have shown to

have the best topological quality than other comparable algorithms (without hole-filling post-processing). The output triangle meshes of other algorithms all contain relatively large topological difference with the original object surface. The power crust algorithm [38] can output a watertight surface mesh but it may need the incorporation of some extra points. Furthermore, it does not produce a triangle mesh interpolating all the given input points. With an additional post-processing step of hole-filling, some algorithms can output a watertight manifold triangle mesh with noticeably lower topological difference, such as tight cocone [2] and umbrella filter [34]. As seen in Table 3.4, however, the proposed UFM algorithm still stands out to have the best topological quality in the reconstructed triangle meshes than these algorithms. This performance can be solely attributed to the fact that once the UFM algorithm converges, all the fully matched umbrellas will guarantee a watertight manifold triangle mesh without the need of hole-filling post-processing. Furthermore, the output of all fully matched umbrellas' facets still guarantees the manifoldness of the final reconstructed triangle mesh with some holes when the UFM algorithm cannot converge. Figure 3.11 shows the different quality of the reconstructed triangle mesh of Casting Die data set from the current algorithm and tight cocone algorithm. Figure 3.11a is a manifold reconstructed triangle mesh from the UFM algorithm and Figure 3.11b is a non-manifold result from the tight cocone algorithm. Evidently, the reconstructed triangle mesh surface in Figure 3.11b includes a large topological error (with ΔF being 6901) and can hardly be regarded as an input of the subsequent mesh geometry processing.



(a) from UFM algorithm (b) from tight cocone algorithm

Figure 3.11 Comparison of reconstructed mesh of the Casting Die data set

It should be noted, however, that for some data sets like the Casting Die, the UFM algorithm cannot converge well (fully matched umbrellas for some points in the data set cannot be established). This happens when the distribution of the data points is highly non-uniform and/or the original object surface is highly irregular. As shown in Figure 3.12, the point density appears to be inadequate in the failure areas that the associated data points cannot unambiguously represent the surface geometry in those areas of the Casting Die. Consequently, the reconstructed triangle mesh for the Casting Die is not topologically correct (with ΔF being 219) and the umbrella matching percentage only reaches 99.7%. The holes shown in Figure 3.12 indicate the areas where fully matched umbrella facets cannot be found. Additionally, some shape deviations can be seen between the reconstructed watertight manifold surface and the original object surface. As the reconstructed watertight manifold triangle-mesh surface of the Oilpmp data set demonstrated in Figure 3.13, main shape deviations come from the failure of its sharp features reconstruction.

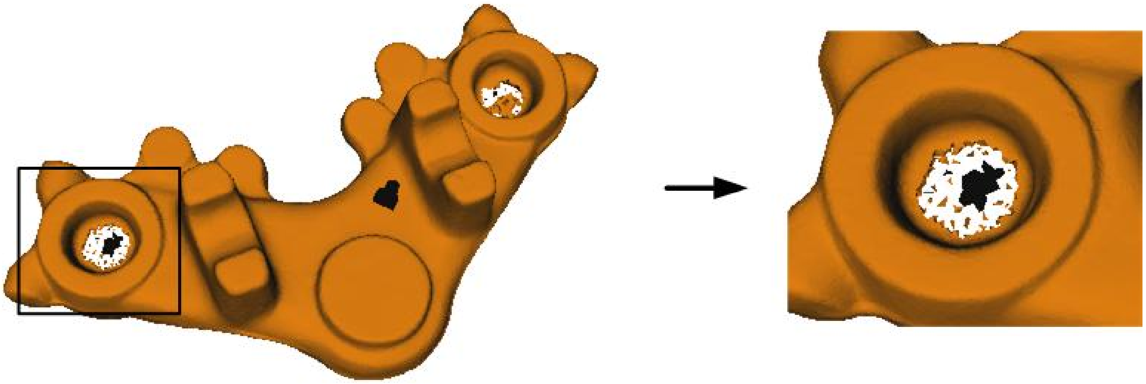


Figure 3.12 Convergence problem in highly non-uniform and under-sampled data

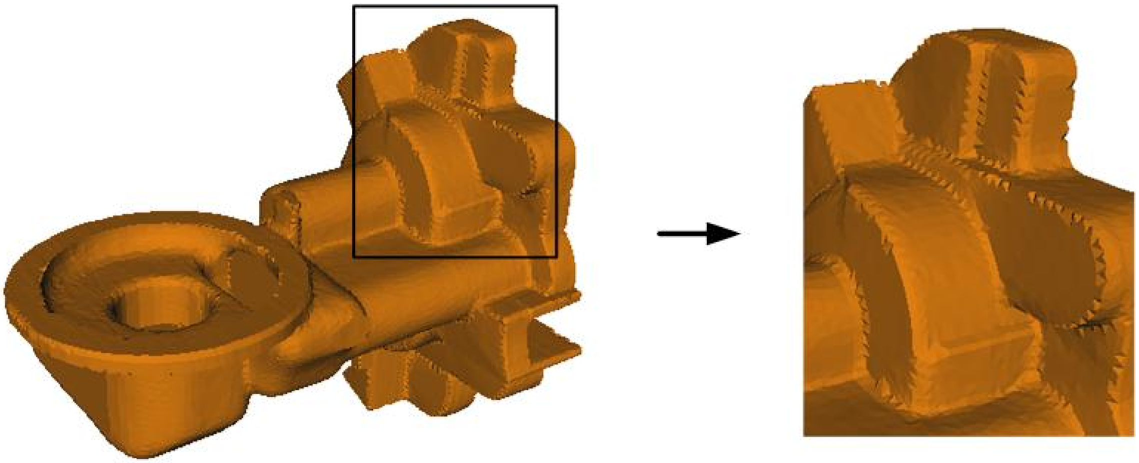


Figure 3.13 Shape deviations in the sharp features

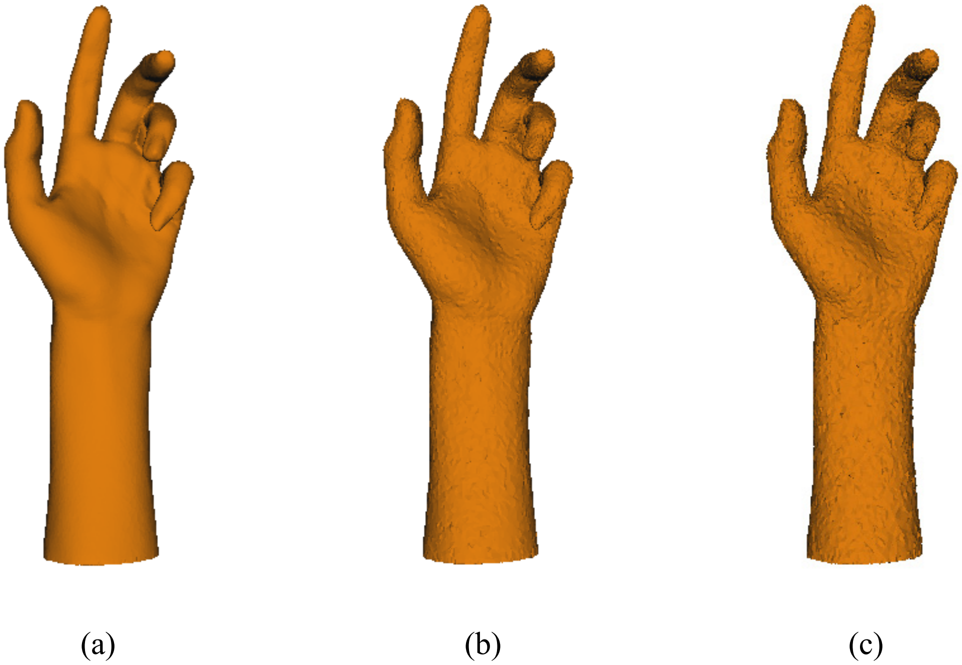

(a) (b) (c)

Figure 3.14 Performance of the UFM algorithm with increasing noise

All Delaunay-based mesh reconstruction methods are sensitive to noise in the point cloud data. How the UFM algorithm handles noise is shown in Figure 3.14. The distribution of the Hand data set is non-uniform. The increasing noise is created by perturbing the points randomly within a small sphere around each point. Figure 3.14a is the reconstructed surface from Hand data set without noise, which is a watertight manifold interpolation surface. Figure 3.14b demonstrates a reconstructed surface from Hand data set with little noise and it remains a watertight manifold interpolation surface. Figure 3.14c shows the reconstructed surface of Hand data set with more noise and the result is not a watertight surface (with few small holes) again. However, for this very noisy point cloud, the reconstructed surface remains a manifold interpolation surface.

Consequently, the proposed UFM algorithm can work well in the low-noise point cloud data and always output a manifold interpolation surface in any case.

3.5. Concluding Remarks

A new and effective algorithm, named as the Umbrella Facet Matching (UFM) algorithm, has been presented in this chapter to generate a watertight manifold triangle mesh from a point cloud. The generated mesh will interpolate all the given data points without either the need of hole-filling post-processing or the need to add extra points. The triangles in the generated mesh are selected from the set of Delaunay triangles at every point. Although the involved Delaunay triangulation computation was generally considered time-consuming in the past, it is not a practical concern anymore thanks to recent advances in the computing power and further improvement in the Delaunay triangulation algorithm. In particular, the UFM algorithm only requires a one-time Delaunay triangulation computation to establish the set of Delaunay triangles at every point.

Although the UFM algorithm cannot guarantee convergence in theory, implementation results have shown that the algorithm in general converges well and all successfully reconstructed meshes are homeomorphic to the original object surfaces. As discussed in Section 3.4, the UFM algorithm still has trouble to converge highly non-uniform and/or under-sampled point cloud data. For future improvement, geometric heuristics may be introduced to control the shape of the umbrella at every point for better convergence. Also, a good reconstructed triangle mesh should match the original object surface with respect to topological equivalence (homeomorphism) as well as shape/feature approximation. In this aspect, many shape deviations can be seen between the reconstructed mesh and the original object surface. Specifically, some sharp corner features cannot be correctly reconstructed. Active research in minimizing the shape deviations will be reported later and an improved solution is introduced in chapter 5.

4 NORMAL VECTOR ESTIMATION FOR POINT CLOUD DATA

4.1. Introduction

Accurate and reliable estimation of normal vector of the point cloud data is of practical importance in computer-aided design and inspection applications, as theoretical or digital model of a given physical object in real-world may not be always available. For example, surface reconstruction from a point cloud data with reliable normal vectors is a much easier problem than surface reconstruction from points set alone. In some surface reconstruction algorithms, the approximation quality of the reconstructed surface heavily relies on how well the estimated normal vectors of point cloud data reflect the true normal vectors of scanned physical object. In fact, normal vectors estimation is often the every first step in surface reconstruction algorithms. This is not only true for Delaunay-based and region growing approaches [24, 37, 42, 48], but also for implicit surface approaches [11, 18, 22, 49-51]. Many other applications often require accurate estimated normal vectors of the point cloud data as well, such as segmentation of the point cloud [52-56] and point-based surface rendering [10, 49, 57].

There have been many proposals for normal vectors estimation algorithms for point cloud data. These algorithms mainly fall into two dominating categories [58]: numerical optimization approach based on the plane or parametric surface fitting technique [11, 59, 60] and combinatorial estimation approach based on geometric analysis of Voronoi diagram/Delaunay triangulation [37, 42, 48, 51]. In any case, general estimation procedure of normal vector in these approaches usually includes two main steps. The first step is to identify a reliable local neighborhood for each point in the point cloud data. The second step is the calculation and orientation of desired normal vector at each point against points in its local neighborhood. More details on normal vector estimation and main features of the proposed method are outlined in the following sections

4.2. Reliable Neighborhood Identification

Essentially, normal vector is a local geometric property of a 2-dimensional surface and specific to each given point. Therefore, reliable estimation of the normal vector at each point in a point cloud data heavily depends on the positive identification of its valid neighboring points in the neighborhood. Choosing too many neighboring points for normal vector estimation can lead large deviations of the estimated normal vectors, especially in the region adjacent to sharp features. Too few neighboring points chosen may result in insufficiency in representing local geometry. A well estimated normal vector is a significant step towards correct reconstruction of sharp features in the original model surface.

4.2.1. Existing Approaches

The approach first used by Hoppe et al. [11] in the context of surface reconstruction is to find the *k*-nearest neighbors of given point p, the set denoted as $N_k(p)$, and take the normal of the least squares best-fitting plane to $N_k(p)$ as the surface normal at p. This normal vector algorithm could be called Plane Fitting (PF) method [58]. These *k*-nearest neighbors $N_k(p)$ can also be used to fit the local quadric surface

[54] for normal vector estimation as well. Pauly et al. [60] improved the PF method by adding different weights depends on the distance between neighboring points and the estimated point. If the distance of a neighboring point from estimated point p is smaller, the larger weight is assigned to the neighboring point. They computed a local reference plane $N^T x = D$ at point p by minimizing the weighted sum of squared distance $\sum_{i=1}^{k}(N^T p_i - D)^2 \varphi(\|p_i - p\|)$. In their computation, the weighting function $\varphi()$ is used to control the characteristics of the surface. Typically, a Gaussian function $\varphi(\|p_i - p\|) = \exp(-\|p_i - p\|^2 / \sigma^2)$ is chosen, where σ is a global scale parameter that determines the characteristic size of the resulting surface and chosen to be one third the square distance between p and p_k (farthest point in $N_k(p)$). This improved PF method could be called Weighted Plane Fitting (WPF) method [58]. Additionally, all points within a fixed or adaptive distance r of point p are often used to choose the neighboring points of point p. Mitra et al. [59] proposed a plane fitting method based on an adaptive distance r to estimate the normal vector for point cloud data. The main problem of *k*-nearest neighbors is bias problem. When the distribution of $N_k(p)$ is non-uniform, the chosen $N_k(p)$ at each point can not provide good enough local geometric information for its normal vector estimation.

An alternative way is to construct a polygonal mesh surface for a point cloud data and identify neighboring points at each point from its local mesh neighbors. The typical application is the combinatorial estimation approach based on geometric analysis of Voronoi diagram/Delaunay triangulation. As discussed previously, Voronoi diagram and Delaunay triangulation can provide a powerful way to approximate the neighbourhood at each point in a point cloud data. Hence the local mesh neighbors at each point always can be extracted from Delaunay triangles set incident to each point. Voronoi diagram and Delaunay triangulation are a global geometric data structure from computational geometry and can be built from any point cloud data with arbitrary distribution and density. If the reliable local meshes at each point can be extracted as the local mesh neighbors, estimated normal vectors depending on these mesh neighbors can be more accurate.

Nowadays, different methods have been proposed by researchers to extract desired Delaunay triangles incident to a point p to identify the local Delaunay-triangle mesh neighbors of p from the global Delaunay triangulation of P. Adamy et al. [34] proposed a method to build an umbrella at each point from their Gabriel subset of Delaunay triangles set. The building of an umbrella is an incrementally adding triangle processing based on the proposed concept of λ-interval, which often requires a manifold post-processing. The resulting umbrella in fact is a kind of local mesh neighbors. Ouyang et al. [61] proposed a method based on region growing to build the local Voronoi mesh neighbors at each point, which is similar to the BPA algorithm. Depending on the identified “good” neighboring points from the built local Voronoi mesh, a novel quadric curve fitting algorithm was provided to calculate the desired normal vector. A benefit of their algorithms is that the required number of neighboring points could be only three. Their normal vector estimation algorithm is called local Voronoi mesh method or LVM in short. However, the estimated normal vectors based on local mesh neighbors heavily depend on the quality of built local meshes. All methods mentioned above did not provide how reliable these built local (Voronoi/Delaunay) meshes are.

4.2.2. Local Mesh Neighborhood

In fact, the neighborhood of each point in a point cloud data P has been widely studied in the computational geometry community and many efficient algorithms exist that solve a number of geometric problems [62], such as closest points, all nearest neighbors, Euclidean minimum spanning tree (EMST), etc. These problems can be efficiently solved by graphs representation in which pairs of points that are linked by an edge. Many geometric properties of these graphs are benefit to surface reconstruction. In fact, they have been used by some researchers as an initial step in surface reconstruction, such as the Euclidean minimum spanning tree ($EMST(P)$), Gabriel graph ($GG(P)$) and Delaunay triangulation ($D(P)$). If the edges set in Delaunay triangulation is denoted by $ED(P)$, the relationship of these graphs of a point cloud P can be expressed as following Equation 4.1:

$$EMST(P) \subseteq GG(P) \subseteq ED(P) \tag{4.1}$$

In some curve or surface reconstruction algorithms, as an initial graph, EMST can guarantee that the resulting edges are the shortest possible. Therefore, close points in the point cloud data are likely to be linked in the graph, which is helpful to reconstruct the desired curve or surface. The Gabriel graph has also been used for curve or surface reconstruction. It gives clue about best interconnection among points when used for the reconstructing the boundary of a 2D point cloud data [45, 63].

In Delaunay-based surface reconstruction approaches, for a point cloud P in Euclidean space R^3, the desired reconstructed triangle-mesh surface is extracted from the Delaunay triangles set $D(P)$ of a point cloud P. The Gabriel triangles set $GT(P)$, a subset of $D(P)$, is often applied at the initial step to help seek the candidate Delaunay triangles for reconstructing the desired triangle-mesh surface [34, 46, 64]. For a point v in P, the relationship $GT(P) \subseteq D(P)$ could also be locally described in Equation 4.2:

$$GT_v \subseteq DT_v \tag{4.2}$$

However, the desired correct triangle mesh can not be extracted from Gabriel triangles set alone, although these Gabriel triangles have a high probability of being close to the original surface. Sometimes, all Gabriel triangles (GT_v) at a point v is not a pocket triangles set (Figure 3.2c or Figure 3.2d), which means a full umbrella can not be extracted from GT_v. Adamy et al. [34] noticed the problem and try to employ a post-processing to fix these missing triangles. In the algorithm presented in chapter 3, a full umbrella can be guaranteed to be generated at each point. Based on a full umbrella at each point, a novel evaluation methodology of these umbrellas matching is proposed, which provides a refined way of identifying reliable local Delaunay triangulation mesh neighbors at each point. This is a kind of subset of Delaunay triangles set which is based on the full umbrella, a local manifold mesh incident to a point. The refined relationships can be locally described the following Equation 4.3:

$$U(\bar{f})_v \subseteq U(f)_v \subseteq DT(U)_v \subseteq DT_v \tag{4.3}$$

where DT_v denotes all Delaunay triangles incident to point v, $DT(U)_v$ all umbrella Delaunay-triangles incident to point v, $U(f)_v$ triangular facet set in the umbrella at point v, $U(\bar{f})_v$ all fully matched triangular facets in the umbrella at point v.

A simple example is illustrated in Figure 4.1. Three input points v_1, v_2 and v_3 and their umbrellas U_{v_1} , U_{v_2} and U_{v_3} are drawn in Figure 4.1, where $U_{v_1} = \{f_1, f_2, f_3, f_4, f_5, f_6\}$, $U_{v_2} = \{f_6, f_7, f_8, f_9\}$ and $U_{v_3} = \{f_5, f_6, f_9, f_{10}, f_{11}\}$. According to the evaluation methods proposed in chapter 3, in these Delaunay-triangles set incident to v_1, v_2 and v_3 , there exist one fully matched triangle f_6 ($M_{f_6} = 3$). The matching index of triangle f_5 and f_9 is two ($M_{f_5} = 2$ and $M_{f_9} = 2$). The matching index of other triangles is one. In Equation 4.3, for point v_1, we get a refined way to identify the local mesh neighbors: $U(\bar{f})_{v_1} = \{f_6\}$, $U(f)_{v_1} = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ and $DT(U)_{v_1} = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7\}$. All neighboring points in $DT(U)_{v_1}$ of point v_1 are drawn in red color dot in Figure 4.1.

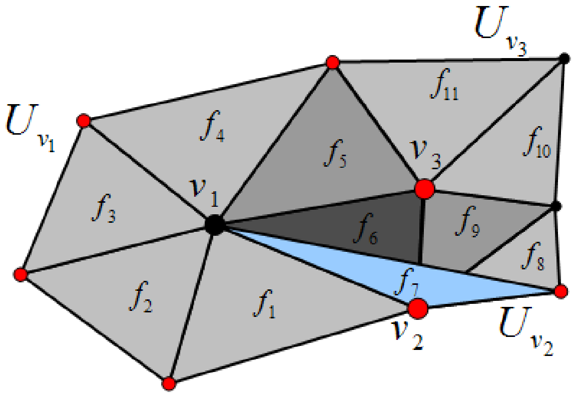
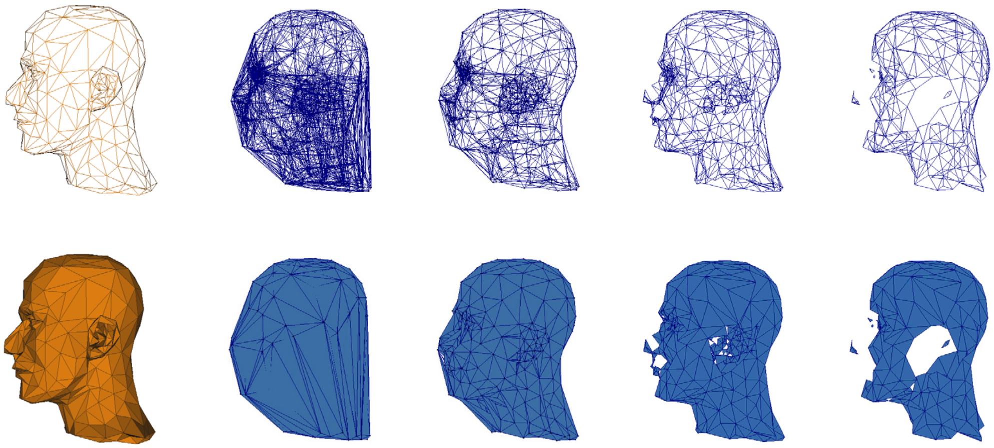


Figure 4.1 An example of local mesh neighbors based on umbrella matching

Figure 4.2 demonstrates the comparison of global mesh composed by these different local mesh neighbors. Figure 4.2a is the original triangle mesh surface for point cloud P, Figure 4.2b is the global Delaunay triangulation of P ($D(P)$) and Figure 4.2c is the Gabriel triangles subset ($GT(P)$). Figure 4.2d and Figure 4.2e shows the $U(f)(P)$ and $U(\bar{f})(P)$ subsets respectively based on the local umbrellas, which are derived from the evaluation methods proposed in chapter 3. Evidently, the triangle mesh in Figure 4.2d is the best approximation for the original triangle mesh surface and Figure 4.2e perhaps can provide more accurate local mesh neighborhood for some points but missing too many regions. That also implies that the identified local mesh neighbors in Figure 4.2d can supply the optimum local neighborhood geometric information for the normal vector estimation at each point. Details of normal vector computation based on the refined local mesh neighbors are described in the following section.



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

Figure 4.2 Global meshes comparison based on different local mesh neighbors

4.3. Computational Procedure

The well-known Euler Equation 4.4 [65] describes the relationship between the numbers of vertices V, edges E and faces F in a closed 2-manifold polygonal mesh:

$$V - E + F = 2(1 - G) \qquad \textbf{(4.4)}$$

where G is the genus of an object and intuitively represents the number of through-holes of the object. Because the genus of typical meshes is small compared to the numbers of mesh elements, the right-hand side of Equation 4.4 can be assumed to be close to zero. For a closed 2-manifold triangle mesh, each triangle is bounded by three edges and each edge is incident to two triangles, the following triangle mesh statistics can be driven [1]:

- The number of edges is three times the number of vertices: $E \approx 3V$
- The number of triangles is twice the number of vertices: $F \approx 2V$
- The average number of edges or triangles incident a vertex is 6

The average number of incident edges for point v is also called vertex average degree or valence. As discussed previously, the triangles in $U(\overline{f})_v$ could provide the more accurate and reliable local neighborhood geometric information. If the number of triangles in $U(\overline{f})_v$ at point v is more than half of average valence of point v, a normal vector estimation based on the normals of these incident triangles can become a better calculation method for point v. Therefore, a combinatorial normal vector calculation is proposed in this chapter, which relies on the refined local mesh neighbors identification described in the previous section. When the number of the fully matched umbrella facets ($U(\overline{f})_v$) at point v is no less than three, the normal vector N_v calculation adopts a weighted average of the normal vectors of triangles in the $U(f)_v$, as described in the Equation 4.5:

$$N_v = \frac{\sum_{i=1}^{k} w_i n_i}{\sum_{i=1}^{k} w_i} \quad \textbf{(4.5)}$$

where n_i and w_i are the normal vector and weight of the ith triangle f_i in $U(f)_v$, respectively. The weight is the combination of angle-based weight and matching index weight, and is defined in the following Equation 4.6.

$$w_i = M_{f_i} \bullet \angle p_i v p_{i+1} = M_{f_i} \bullet \cos^{-1}\left(\frac{\overline{vp_i} \bullet \overline{vp_{i+1}}}{\left|\overline{vp_i}\right| \left|\overline{vp_{i+1}}\right|} \right) \quad \textbf{(4.6)}$$

where $\Delta p_i v p_{i+1}$ is the ith triangle f_i in $U(f)_v$, M_{f_i} denotes the matching index of f_i and $M_{f_i} = \{1, 2, 3\}$.

When the number of the fully matched umbrella facets ($U(\bar{f})_v$) at point v is less than three, the normal vector N_v calculation adopts the same weighted plane fitting technique (Equation 4.7) for the all neighboring points in $DT(U)_v$ local meshes set, as set out in the WPF method [60].

$$\left\{ \begin{array}{l} \sum_{i=1}^{k} (N^T p_i - D)^2 \varphi(\|p_i - p\|) \\ \varphi(\|p_i - p\|) = \exp(-\|p_i - p\|^2 / \sigma^2) \end{array} \right. \quad \textbf{(4.7)}$$

Whatever method is adopted to compute normal vectors, the estimated normal vector should always be expected to be oriented consistently with each other on the correct side of the surface (inside or outside). Finding a globally consistent orientation for estimated normal vectors is not easy especially for the point cloud data with low density, noise or sharp features. In the normal vector estimation algorithm presented in this chapter, a popular method proposed by Hoppe et al. [11] is employed to orientate all estimated normal vectors. Considering that normal vector at a point on a surface is also the normal vector of its fitted tangent plane, which can serve as the local linear approximation to the surface, the algorithm propagates the tangent plane's normal direction based on a constructed Riemannian Graph. When the point cloud data is sufficiently dense, the propagation method can successfully orientate all estimated normal vectors. The flowchart of the proposed normal vector estimation algorithm is shown in Figure 4.3.

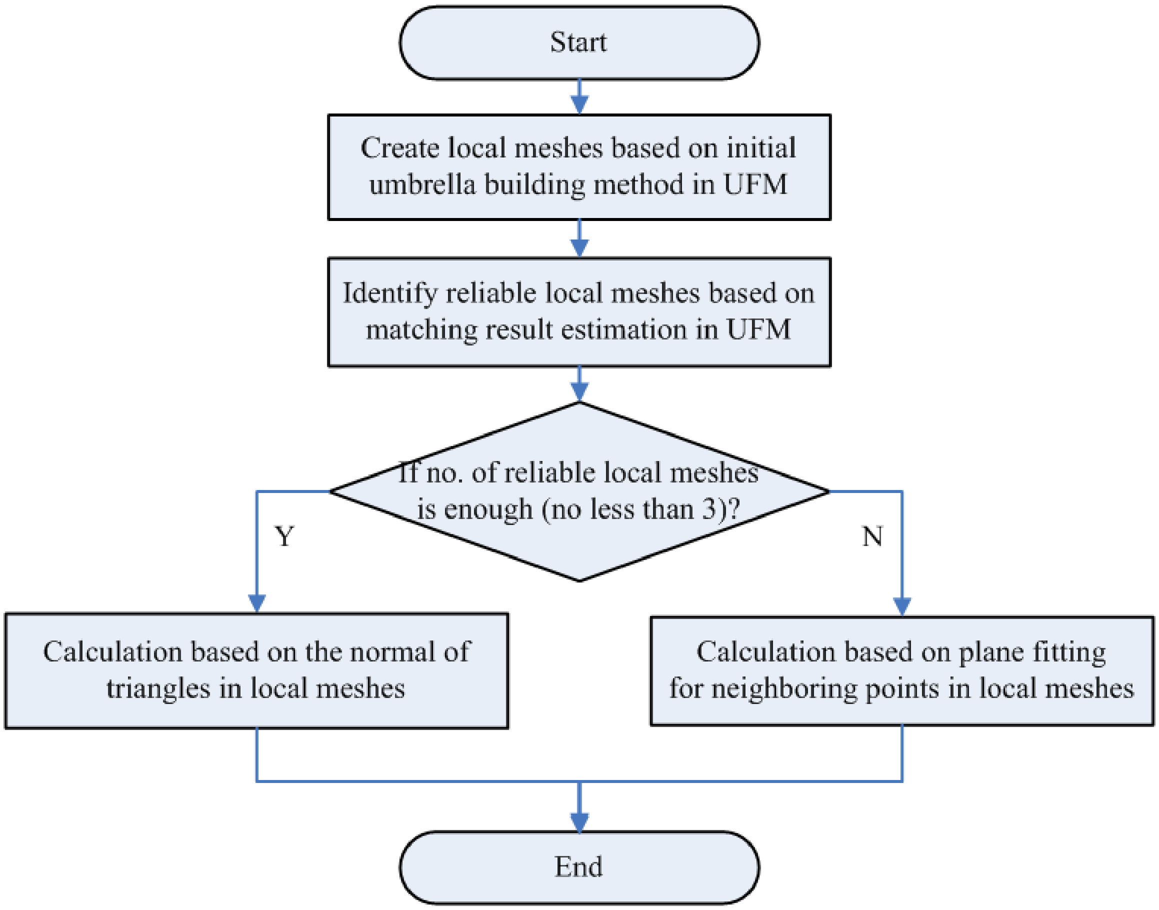

Start
Create local meshes based on initial umbrella building method in UFM
Identify reliable local meshes based on matching result estimation in UFM
If no. of reliable local meshes is enough (no less than 3)?
Y
N
Calculation based on the normal of triangles in local meshes
Calculation based on plane fitting for neighboring points in local meshes
End

Figure 4.3 Normal vector estimation flowchart

4.4. Implementation Results and Discussion

Multiple case studies have been performed and analyzed to validate the performance of the proposed normal vector estimation method in this chapter. The plane fitting (PF) algorithm by Hoppe et al. [11], weighted plane fitting (WPF) algorithm by Pauly et al. [60] and the local Voronoi mesh (LVM) algorithm by Ouyang et al. [61] are chosen for comparison with our proposed method in the following section.

4.4.1. Case Study Setup

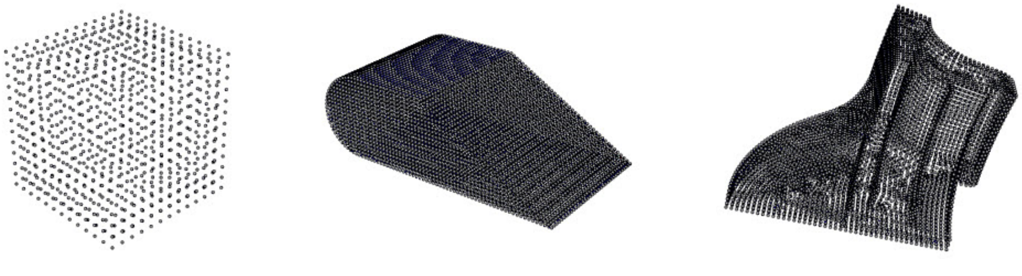
In order to compare the resulting normal vector estimated from different methods, ideally we need to measure the deviation of the estimated normal vectors from the “true" surface normal vectors N'. However for the real point cloud data, the original surface of a model is generally not available. Therefore, three different types of point cloud testing data are designed for the following case studies.

The first type is a simulated point cloud data generated by parametric mathematic functions and its exact normal vector N' can thus be numerical calculated at each point. Two simulated point clouds data from Torus (Figure 4.4a) and Ellipsoid (Figure 4.4b) parametric algebraic surface are generated uniformly for our case studies. As we known, the local surface shape at a point can be approximated by a quadric surface. According to the curvature tensor of each point, the point can be considered as a parabolic, an elliptical or a hyperbolic point [66]. All these three types of points can be found on a Torus surface. The point cloud data from Ellipsoid surface is regarded as a points set with the high curvature region. They are shown in Figure 4.4.

(a) Torus (b) Ellipsoid

Figure 4.4 The first type data for normal vector estimation comparison

The second type is a simulated point cloud data generate from a known uniform-mesh model, which could be a meshed CAD model or ideal model with known correct meshes. The normal vector N' at each point (mesh point) can be estimated from an area-weighted average of the normal vectors of its local incident meshes. Although N' in the second type data is not "true" original surface normal vector like the first type, it is still accurate enough to become a referential normal vector for algorithm comparison as long as the mesh of the designed model is uniform and not too sparse. In our case studies, Cubic (Figure 4.5a), SimulationSolid (Figure 4.5b) and Fandisk (Figure 4.5c) point cloud data belong to the second type data, as shown in Figure 4.5.



(a) Cubic (b) SimulationSolid (c) Fandisk

Figure 4.5 The second type data for normal vector estimation comparison

The third type of testing data is designed for real scanned point cloud data. We reconstruct a desired triangle-mesh surface from the real point cloud data by a known surface reconstruction algorithm called tight cocone [2]. Then, the area-weighted average of the normal vectors of the triangles incident to a point p in this surface is taken as the referential normal vector N' at p. The estimated referential normal vector N' in this third category is not accurate but close enough to analyze the results from different algorithms in real point cloud data.

We define the error e of an estimated normal vector at each point as the angle (in radians) between the referential normal vector N' and the estimated normal vector N, as described in the following Equation 4.8:

$$e = \cos^{-1}\left(\frac{N \bullet N'}{|N||N'|}\right) \tag{4.8}$$

Evidently the smaller the error e, the better the estimated normal vector is.

Table 4.1 Normal vector estimation errors (in radians)

Model name		**Torus**	**Ellipsoid**	**Cubic**	**Simulation-Solid**	**Fandisk**
No. of Points		3600	9950	866	6988	6475
Mean error	Current	0.00105	0.00126	0.03122	0.02245	0.01791
	LVM	0.00593	0.00202	0.07526	0.02374	0.02546
	PF	0.01584	0.00412	0.15641	0.07302	0.13652
	WPF	0.00818	0.00337	0.16784	0.07021	0.13719
RMS error	Current	0.00120	0.00176	0.10691	0.12560	0.07970
	LVM	0.00688	0.00242	0.18131	0.12322	0.08659
	PF	0.01806	0.00573	0.21884	0.18267	0.21557
	WPF	0.01296	0.00654	0.22828	0.17456	0.21145
Standard Deviation	Current	0.00058	0.00174	0.10231	0.12359	0.07766
	LVM	0.00350	0.00133	0.16505	0.12092	0.08277
	PF	0.00868	0.00399	0.15315	0.16745	0.16685
	WPF	0.01006	0.00560	0.15482	0.15983	0.16092
Timing	Current	2.94	5.67	1.44	5.10	4.51
	LVM	3.07	7.12	0.65	5.15	5.08
	PF	0.36	1.34	0.17	0.50	0.55
	WPF	0.51	2.64	0.23	0.58	0.82

4.4.2. Analysis and Comparison

The value of k in $N_k(p)$ for PF and WPF algorithms is assigned 30 and 40 (recommended in PointShop3D [67]) respectively in our case studies. The estimated normal vector errors from different algorithms based on the first and second type simulated point cloud data are shown in Table 4.1. The mean errors, root mean square (RMS) errors, standard deviations and timings are listed for comparison.

Since the testing point cloud data is noise-free, all four methods make good estimations of normal vector as indicated by small mean errors, RMS errors and standard deviations. However, the current algorithm and the LVM algorithm work better than the other two numerical optimization approaches based on the (weighted) plane fitting of $N_k(p)$, which both belong to combinatorial estimation approach based on local mesh neighbors. The algorithm proposed in this chapter demonstrates the minimum normal vector estimation errors. For the further comparison and analysis, their colour maps of normal vector deviation are plotted in Figure 4.6 and Figure 4.7. The blue means the smaller deviation and the red means the larger deviation.

In Figure 4.6, both PF and WPF algorithms generate larger deviation on the top of Torus. The reason behind this is that the center of fitting plane based on $N_k(p)$ at each point in the region generates a large bias with itself. This renders the fitting plane incapable of approximating the desired tangent plane for each point well and leads to larger errors in the final estimated normal vectors in this region than those in other regions. Likewise, for a point at the high curvature region in Ellipsoid, the neighboring points from the point cloud does not lie close to a plane and hence the fitting plane computed by PF or WPF method could not approximate the tangent plane properly at those points.

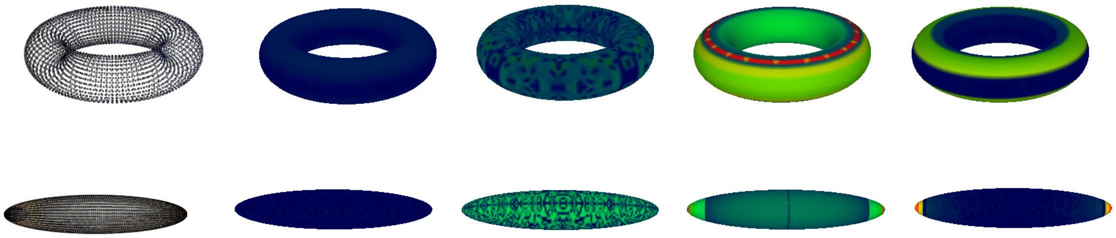

(a) point cloud (b) current algorithm (c) LVM algorithm (d) PF algorithm (e) WPF algorithm

Figure 4.6 Comparison of estimated normals for simulated data

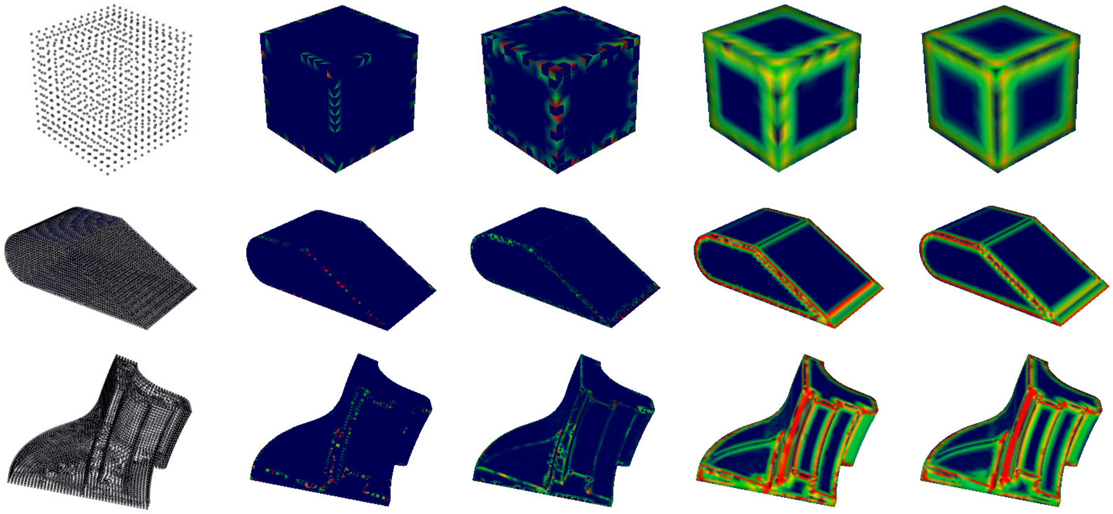
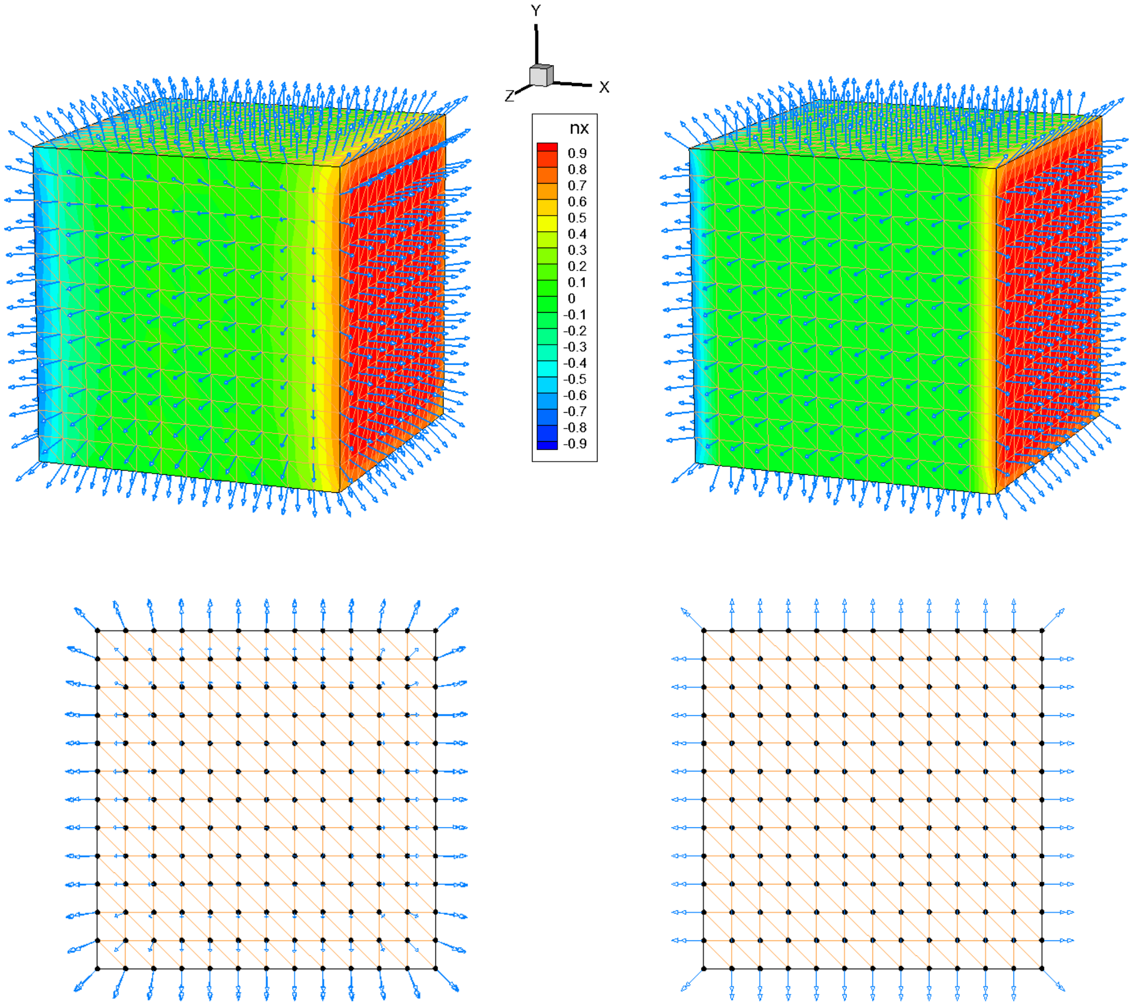

(a) point cloud (b) current algorithm (c) LVM algorithm (d) PF algorithm (e) WPF algorithm

Figure 4.7 Comparison of estimated normals for simulated data with sharp features

For point cloud data with sharp features, such as Cubic, SimulationSolid and Fandisk data set, the current algorithm and the LVM algorithm have the clear advantage in estimating the accurate normal vectors, especially for the regions adjacent to the sharp features. The advantage is attributed to their accurate estimation of local mesh neighbors. Compared to these combinatorial estimation approach based on the local mesh neighbors, the PF and WPF numerical optimization approaches often fail to estimate a proper normal vector for points in the region adjacent to sharp features or with high curvature, as shown in Figure 4.7.

Figure 4.8 gives a more accurate analysis for estimated normal vectors deviation in the region adjacent to sharp features, where a cubic data set is selected for case study. The normal vectors of simulated point cloud data from a cube model are estimated from both the weighted plane fitting algorithm [59] (Figure 4.8a) and the proposed algorithm in this chapter (Figure 4.8b). The top row plots in Figure 4.8a and Figure 4.8b compare 3D color maps of corresponding deviation of estimated normal vector at each point along X axis. At each point its normal vector is marked in red ($nx = 1$), if it orients to X axis; and in blue ($nx = -1$) if it orients to $-X$ direction. For normal vector perpendicular to X axis it is marked in green ($nx = 0$). Evidently, the estimated normal vectors from the proposed algorithm in this chapter demonstrate better results in regions adjacent to sharp features due to its more accurate local Delaunay triangulation mesh neighbors. The current algorithm is pivotal in solving the sharp-feature preservation issue in surface reconstruction.

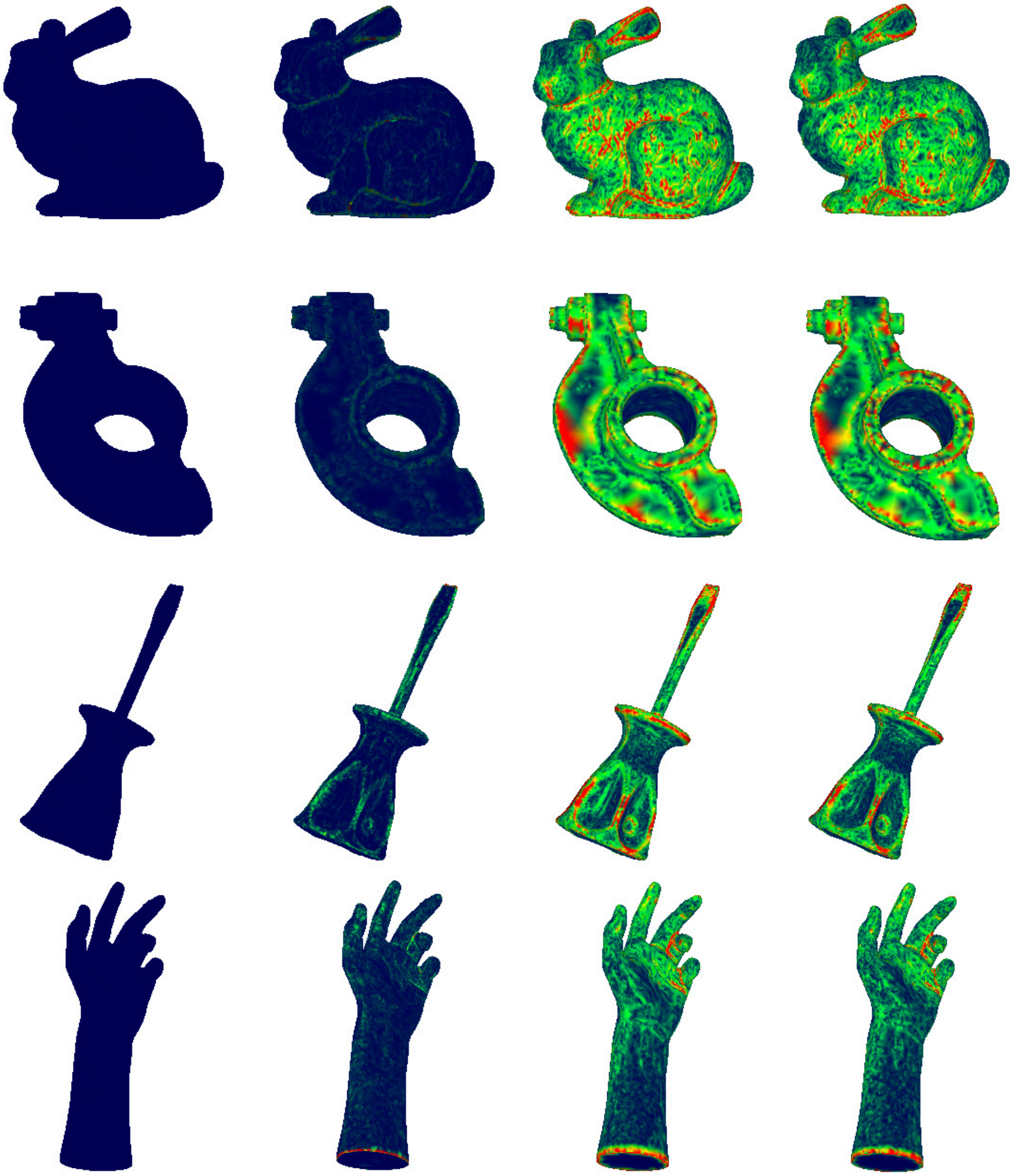


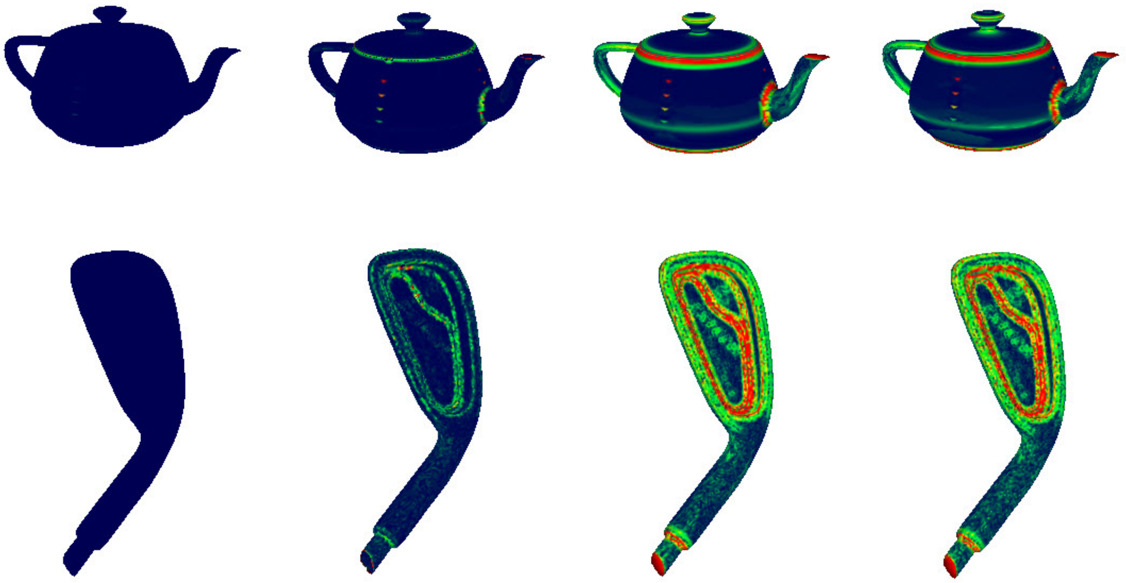
(a) WPF algorithm

(b) current algorithm

Figure 4.8 Comparison of estimated normals for Cubic data

Figure 4.9 demonstrate many case studies on real scanned point cloud data based on their colour maps of the estimated normal vector deviation, as the third type of testing data. The referential normal vector N' at each point is computed through the area-weighted average of the normal vector of incident triangle mesh. The referential original triangle mesh surface is reconstructed by the tight cocone algorithm. Although N' is not the "true" surface normal vector, the results shown in Figure 4.9 provide enough demonstration on the larger errors of normal vectors estimated from PF and WPF algorithms in the region with high curvature or adjacent to the sharp features compared to the combinatorial approach based on local mesh neighbors.





(a) current algorithm

(b) LVM algorithm

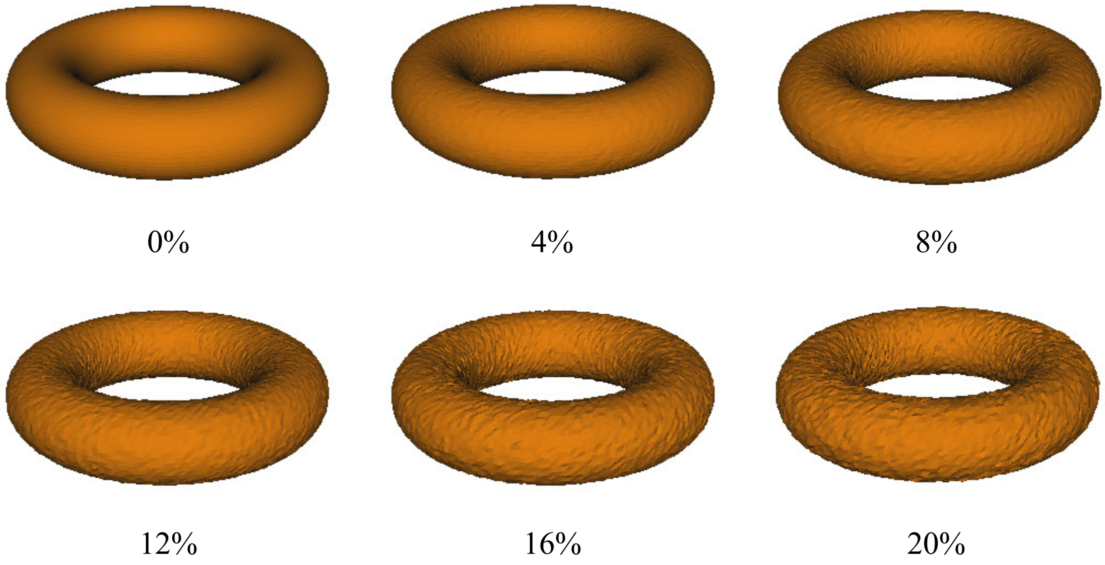
(c) PF algorithm

(d) WPF algorithm

Figure 4.9 Comparison of estimated normals for real point cloud data

4.4.3. Limitation

In our case study for noisy data, we obtain testing subject by adding noise to the original point cloud data. The x, y and z components of the noise are independent and uniformly distributed. The noise level is controlled by a local scale factor. The average distance of a point p to its ten nearest neighbors is chosen as the factor. The point p is thus perturbed by this factor. Six factors 0, 0.04, 0.08, 0.12, 0.16 and 0.2 are considered for this case study. The testing data is from a uniform point cloud data (the second type testing data) from Torus surface with 10000 pts and made noisy by artificial perturbation with six different noise levels, as shown in Figure 4.10.



0% 4% 8%

12% 16% 20%

Figure 4.10 Torus with increasing noise

As discussed above, for a point cloud data with no noise, the proposed algorithm and LVM algorithm both demonstrate better performance for normal vector estimation, especially around high curvature and regions with the sharp feature. However, for a noisy point cloud data, the numerical optimization algorithms based on plane fitting techniques demonstrate their advantages in Figure 4.11. When the noise is low, all four algorithms estimate normal vector well. As the noise level increases, the proposed algorithm and LVM algorithms normal vector estimation become worse.

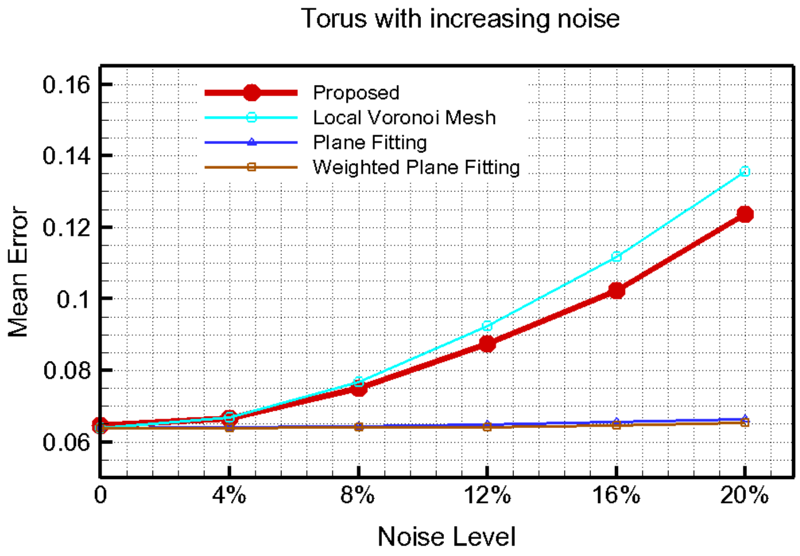


Figure 4.11 Estimated normal errors in different noise level

Therefore, the proposed normal vector estimation algorithm in this chapter is noise sensitive. As we know, increasing neighboring points can decrease the normal vector estimation error in the noisy point cloud data not only for numerical optimization approaches but also for combinatorial estimation approaches. For the local umbrella mesh neighbors incident to point v, there exist a ring of an umbrella. All circumjacent neighborhood vertices $U(p)_v = \{p_1, \dots, p_n\}$ in the umbrella U_v at point v are regarded as the one-ring neighboring points set at point v. If one-ring neighboring points set of all points in the one-ring neighboring points set ($U(p)_v$) are also counted in the neighboring points set for point v (except itself), we term it two-ring neighboring points set at point v. Using two-ring neighboring points set into current algorithm can dramatically reduce estimation error in the noisy point cloud data as demonstrated in Figure 4.12. This makes it possible for users to choose the number of rings according to the noise level of the point cloud data.

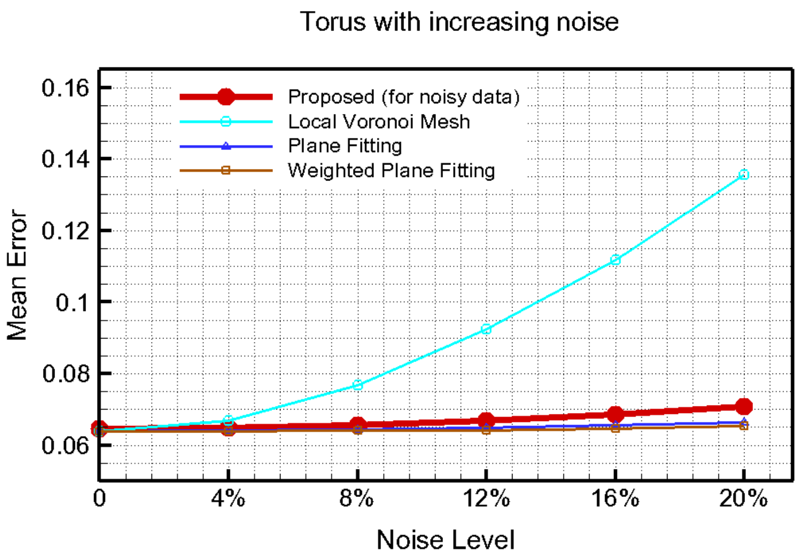


Figure 4.12 Proposed algorithm with two-ring neighbors

4.5. Concluding Remarks

A new combinatorial normal vector estimation method for point cloud data from the local mesh matching results at each point has been proposed in this chapter. Built upon the novel evaluation methodology of local mesh matching proposed in the last chapter, a refined local Delaunay triangulation mesh neighbors at each point can be identified. These well estimated local Delaunay triangulation mesh neighbors make it possible to compute a reliable normal vector at each point, especially for points adjacent to the sharp features. The computation of normal vector at point v is divided into two different matching-index weighted calculations depending on the number of fully matched umbrella facets incident to v ($U(\overline{f})_v$). When the number of $U(\overline{f})_v$ is no less than 3, a weighted calculation based on the normal vectors of all triangles in $U(f)_v$ is adopted. Otherwise, a weighted calculation based on the plane fitting for all neighboring points in $DT(U)_v$ is adopted.

Different from other combinatorial algorithms based on local meshes neighbors (such as LVM algorithm), the proposed algorithm can identify reliable local mesh neighbors depending on matching results estimation in chapter 3, which can help improve the accuracy of estimated normal vector. Comparing with the general numerical optimization approaches, such as plane fitting (PF) and weighted plane fitting (WPF) algorithms, the proposed combinatorial normal vector estimation algorithm yields more accurate result for low-noise or no-noise point cloud data, especially in the region adjacent to the sharp features, though it might be more time-consuming. The proposed normal vector estimation algorithm will be applied in the next chapter to help solve sharp-feature preservation issue. The effectiveness of the proposed algorithm has been demonstrated with both simulated and real-world point cloud data sets. Multiple case studies have been performed and analyzed to validate its performance.

5 NORMAL VECTOR CONE FILTERING FOR SHARP FEATURE RECONSTRUCTION

5.1. Introduction

With recent advancements in optical technologies, 3D laser scanners have dramatically improved both in precision and in affordability. As 3D laser scanning finds its application in many fields such as design, manufacturing, and art, the issue of reliable conversion of the scanned point cloud data into a mathematical surface representation has long been recognized and actively investigated in the surface reconstruction research community. Especially in computer-aided inspection based on ultra-precise 3D laser scanning system, it is considered mission critical that exact sharp features of the original physical object can be reconstructed from the measured point clouds.

The term "feature" has been used and well-defined in many disciplines and applications. For example, in computer-aided design and manufacturing, feature design means directly introducing functional features or manufacturing features into the product model in order to streamline design stage. Feature recognition focuses on extraction of manufacturing or form features from a solid model [68]. For geometric modeling, a free-form feature is defined as a visually prominent characteristic of the shape [69], including but not limited to, sharp edges, ridge lines, valley lines, corners, etc. Evidently, sharp features are of vital importance and typically represent critical sections of the model; thus, exact identification or reconstruction of them is essential not only for quality control in measured point cloud data but also for reverse engineering with surface reconstruction.

Most existing surface reconstruction algorithms discussed in chapter 2.2 yield a smooth triangle-mesh surface from input point clouds data. In general, an extra post-processing algorithm or remeshing process is required to preserve sharp features of the original physical object, such as the combined Sharpen&Bend post-processing [70]. Automatic identification or direct reconstruction of sharp features still remains an open research question in triangle mesh surface reconstruction. As illustrated in Figure 5.1, a typical mechanical part with sharp features can be reconstructed differently from different the reconstruction algorithm. Figure 5.1a shows the point cloud of the well-known Fandisk model and Figure 5.1b shows the output mesh using a typical smooth surface reconstruction algorithm. Figure 5.1c is the reconstructed triangle mesh from the proposed algorithm in this chapter, which preserves sharp features of the Fandisk model well, in comparison with the general method used for Figure 5.1b.

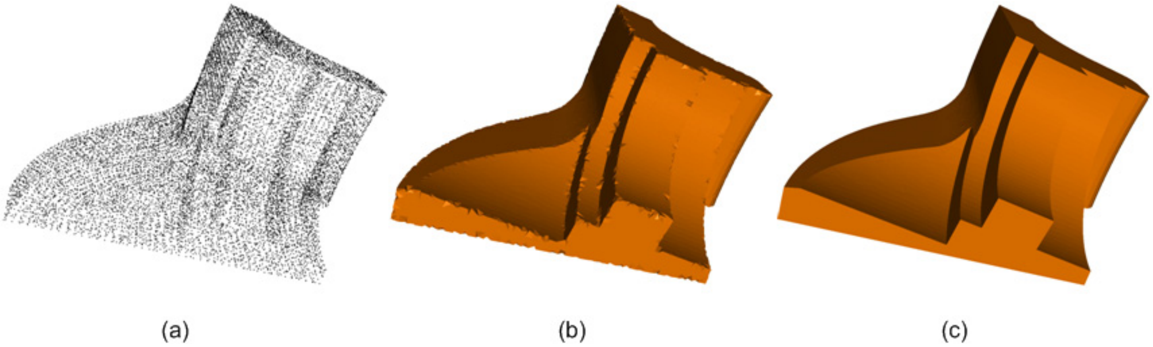


Figure 5.1 An example with sharp features: (a) original point cloud; (b) reconstructed mesh by a general algorithm; and (c) reconstructed mesh by the proposed feature sensitive algorithm

As stated in chapter 2.2, majority of surface reconstruction algorithms developed in the past decades can be classified into three main categories: implicit surface, region growing, and Delaunay-based approaches. The basic idea behind the implicit surface approach is the building of a function in the Euclidean space R^3, from input point cloud, which is formulated to be negative inside the modeled object and positive outside. The output surface can be extracted simply as the zero level-set of the formulated function. This approach has been employed and implemented by many researchers [11-13, 18, 71]. Additionally, Carr et al. [15] and Dinh et al. [14] applied implicit surfaces based on radial basis functions to a number of problems in computer graphics, including surface reconstruction. All these approaches require a post-processing step to compensate the loss of sharp features because of the limitation of the standard marching cubes algorithm [72]. The problem has been addressed in Refs. [73, 74], where the standard marching cubes algorithm is extended or improved in order to preserve sharp features of the original object. Casciola et al. [75] also proposed an anisotropic extension of radial basis functions to reconstruct surface with sharp features. Nevertheless, implicit surface approaches can only produce reconstructed surfaces that approximate the input points rather than interpolate them and this limits their applications. For applications such as computer-aided inspection or reverse engineering, constraining the measured points to be exactly on the reconstructed surface is often mandatory.

The region growing approach first selects a triangle as an initial region and then incrementally grows or expands the boundary of the initial region by adding new triangles until the whole point data set is covered. This approach is very computationally efficient but often requires additional user-specified parameters. For example, both the ball-pivoting algorithm (BPA) by Bernardini et al. [24] and the combinatorial advancing-front algorithm by Huang and Meng [25], are typical region growing approaches. Lin et al. [26] proposed an improved region growing method based on an intrinsic property of the point set. The method attempts to overcome the limitation of user-specified parameters. Other researchers [9, 28] also aimed to apply the region growing method by picking triangles from Delaunay triangles to reconstruct the desired surface. In any rate, it is factual for all existing region growing approaches that a post-processing algorithm is required not only to fill holes but also to rectify the mesh quality and sharp features.

Delaunay triangulation comes from the computational geometry community and can represent the neighborhood of every point in a point set in all relevant directions. Delaunay-based approaches for desired surface mesh reconstruction always start with extraction of a subset of triangles from the complete set of Delaunay triangles. Boissonnat [30] first proposed a Delaunay-based surface reconstruction algorithm that removed tetrahedral and triangles from the set of Delaunay triangles according to certain geometric rules. By now more and more algorithms based on Delaunay triangulation have been proposed, such as the well known alpha shape algorithm by Edelsbrunner et al. [31], the crust and power crust algorithm by Amenta et al. [36, 38], the cocone and tight cocone algorithm by Dey et al. [2, 40], and the umbrella filter algorithm by Adamy et al. [34]. In the author’s previous work [64] described in chapter 3, an umbrella facet matching (UFM) algorithm based on Delaunay triangulation has been proposed to reconstruct a watertight manifold triangle-mesh surface by interpolating all the input points. Among these algorithms, the power crust algorithm [38] based on the weighted Voronoi diagram of the poles (Voronoi vertices) is capable of reconstructing sharp features but resulting reconstructed mesh is not guaranteed to pass through all input points and generate a triangle-mesh surface. To preserve sharp features, most of the existing Delaunay-based algorithms still require an additional post-processing step. The tight cocone algorithm [2] is effectively a post-processing algorithm based on the cocone algorithm [40], which can reconstruct a watertight interpolation triangle-mesh surface with sharp features. Some researchers, such as Kuo et al. [29], proposed a combinatorial approach combining region growing and Delaunay-based methods to reconstruct surface with sharp features. In their work, a region growing algorithm is used for the smooth region and a Delaunay-based algorithm using the poles is applied for the sharp region. The output triangle meshes of this method, however, cannot guarantee to pass through all input points either. Furthermore, a user-specified parameter is required for reliably identifying sharp regions in the reported algorithm.

5.2. Relevant Techniques

In the past, feature identification techniques are applied both to reconstruct surface with sharp features, and to extract sharp features directly from point clouds. For these applications, specific geometric criteria are always employed for the sharp feature identification, such as the curvature extremum, normal vector deviation, and fitted error of a local least-squares plane.

5.2.1. Feature Extraction from Point Clouds

In the absence of connectivity and normal information, feature extraction from input point clouds is not always straightforward. Neighbor graph is often computed as an initial tool to estimate neighborhood geometric information of each input point. Gumhold et al. [76] employed both the Delaunay filtering and the Riemannian graph to deal with noise-free and noisy data set. Furthermore, they conducted numerical analysis based on curvature extremum to compute weights that mark points as potential creases, boundaries, or corners. Finally, the marked points were connected by a minimum spanning tree and fitted into curves to approximate sharp edges. Demarsin et al. [77] computed the normal of each point using principal component analysis and segmented the points into groups based on the normal variation in local neighborhoods. Song et al. [78] proposed a new criterion based on the extrapolated normal vector at each point, named as incompatibility, and attempted to autonomously detect the sharp features in a point cloud data set based on statistical principles. Fleishman et al. [79] and Öztireli et al. [80] applied robust statistics in moving least-squares (MLS) fitting for surface reconstruction and representation separately. All these techniques can extract the sharp features directly from input point clouds by the neighborhood of each point; nevertheless, their accuracy depends on the sampling condition and the neighborhood selection.

5.2.2. Feature Identification from Meshes

Many researchers have investigated the sharp feature identification issue in mesh models. Mencl and Müller [81] proposed a graph-based surface reconstruction algorithm which can deal with varying point density and high surface curvature. They employed a criterion of normal vector deviation based on the dihedral angles of the incident facets. Hubeli et al. [82] also defined some classification operator by using normal vector deviation analysis in their work. Watanabe et al. [83] used discrete differential geometry methods to estimate the mean and Gaussian curvatures. Attene et al. [70] first identified chamfer triangles from a reconstructed triangle mesh based on dihedral angles. New vertices to subdivide the chamfer triangles were inserted in order to recover the sharp feature. Other researchers employing the Delaunay-based approach [2, 29, 38] used the poles (specific Voronoi vertices) as a tool to approximate the surface normal and to subsequently identify sharp features for the reconstructed surface. However, most of the feature identification techniques based on mesh require a reconstructed mesh as the input. The subsequent sharp feature identification depends not only on the quality of the previous reconstructed surface, but also on user-specified parameters.

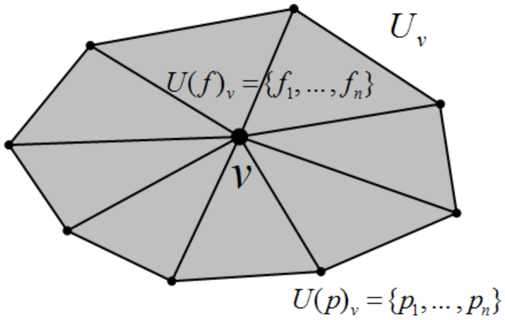

U_v
$U(f)_v = \{f_1, \dots, f_n\}$
v
$U(p)_v = \{p_1, \dots, p_n\}$

Figure 5.2 Umbrella U_v at point v

5.3. Overview of the Proposed Method

This chapter presents a new feature sensitive triangle mesh reconstruction method by analyzing dependable geometric information in the neighborhood of each input point. The neighborhood of each input point is derived from the matching results of the local umbrella mesh constructed at each point. The umbrella is a local 2-dimensional manifold triangular mesh set extracted from a Delaunay triangles set. As shown in Figure 5.2, an umbrella U_v incident to a point v includes a center vertex v, a triangular facet set $U(f)_v = \{f_1, \dots, f_n\}$ and a circumjacent neighborhood vertex set $U(p)_v = \{p_1, \dots, p_n\}$. The evaluation of these matching results is based on the authors' previous work [64], named as the umbrella facet matching (UFM) algorithm. Reliable geometric information in the neighborhood of each point can be well represented by these umbrellas with the current matching results, which is in effect a form of the refined neighborhood graph.

The sharp feature sometimes is considered to be shaped through the intersection of some relatively flat local neighborhood patches. The central idea of the proposed algorithm is to seek reliable local umbrella meshes with “good” flatness in the adjacent region of sharp features, which can be pushed to build the possible relevant sharp features. A novel flatness sensitive filter, referred to as the normal vector cone (NVC) filter in the present work, is introduced and used to seek reliable adjacent umbrellas with “good” flatness by analyzing the neighborhood geometric information of the relevant sharp features. A global flatness parameter based on the dihedral angles is introduced to evaluate the quality of the flatness of an umbrella. The successful configuration of an umbrella with “good” flatness close to the desired sharp feature can help preserve the sharp features in the reconstructed triangle mesh. By resorting to the same unified multi-level priority queuing mechanism in UFM algorithm, our aim is to automatically and reliably reconstruct a watertight manifold triangle mesh with sharp features with a progressive reconstruction process. The reconstructed triangle mesh will be able to preserve all sharp features well and pass through all the original input points without adding or removing any points.

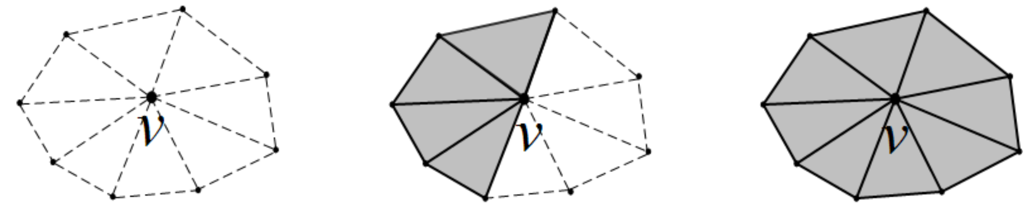
The reminder of this chapter is organized as follows: in the next section, details of the proposed feature sensitive algorithm based on multi-level priority queuing mechanism are described. The elaboration includes a brief report on the basic idea of priority queuing in the UFM algorithm and a detailed description of the normal vector cone filter. Then, further discussion on the presented algorithm is provided with some typical experimental results and analysis. The conclusions and future work are addressed in the last section.

5.4. Feature Sensitive Umbrella Update

As described in the UFM algorithm [64], an umbrella is a local 2-dimensional manifold triangle mesh set at an input point and extracted from a well-known geometric data structure: the Delaunay triangles set or its Gabriel subset. Building an umbrella at a point is essentially a process that successively removes all redundant (non-manifold) Delaunay triangles incident to the point according to a priority queue from its Delaunay triangles set. The remaining triangular facets then correctly constitute a manifold umbrella. It is evident that different priority queues will lead to different umbrellas. In order to get a fully matched umbrella at each point, the corresponding priority queue may be attained by progressively updating the priority queue based on an evaluation of the matching results of all the umbrella facets incident to the point. For an existing umbrella, the matching results of its triangular facets can be evaluated and then used to establish an updated priority queue. An updated priority queue then produces an updated umbrella. This process repeats itself until a fully matched umbrella is found. More details of the progressive meshing process are explored in the following section.

5.4.1. Priority Queuing Based on Matching Results

There are two separate parameters indicating the matching result of the local umbrella constructed at each input point. The baseline matching result is quantified by a absolute matching index M_f, and the refined matching result is quantified by a relative matching index $M_{f(v)}$. Both are designed to evaluate how much an umbrella overlaps with its neighboring umbrellas. If an umbrella triangular facet f is included in all of the three umbrellas of its three vertices, that facet is considered a matched facet ($M_f = 3$), indicated by the shadowed triangles in Figure 5.3. Otherwise, the facet is not a matched facet ($M_f < 3$), shown by the dashed triangles in Figure 5.3.



(a) $\widetilde{U}_v$ void matched umbrella

(b) $\bar{U}'_v$ partially matched umbrella

(c) $\bar{U}_v$ fully matched umbrella

Figure 5.3 Three stages of building a matched umbrella at point v

In the remainder of this chapter, if all the triangular facets of an umbrella at point v are matched facets, the umbrella is called the fully matched umbrella $\bar{U}_v$, as shown in Figure 5.3c. The partially matched umbrella $\bar{U}'_v$, shown in Figure 5.3b, represents an umbrella at point v where only a portion of the umbrella triangular facets are matched facets. If there does not exist any matched facet, the umbrella is named a void matched umbrella $\widetilde{U}_v$ as shown in Figure 5.3a. In fact, Figure 5.3 demonstrates three sequential stages in the process of seeking a matched umbrella at point v.

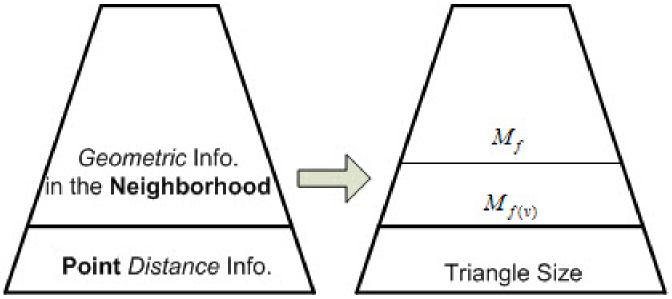


Figure 5.4 Multi-level inheritance priority queuing

In order to construct the desired fully matched umbrella at every point, the priority queuing mechanism with three-level inheritance is introduced in the UFM algorithm. A sub-level always inherits queuing in the super-level. This implies that primary queuing rules should always be placed at the superior levels. As shown in Figure 5.4, the third (bottom) level queuing rule is based on the Delaunay triangle size information which relates to distance information among input points. Other priority queuing level may be unavailable, but the bottom level queuing rule is always available for an input point. This is particularly true for the first umbrella built in the initial stage. The second (middle) level queuing rule is then based on the refined matching result $M_{f(v)}$. The first (top) level queuing rule is based on the basic matching result M_f. In a nutshell, the three-level inheritance priority queuing can extract the required matching results and relevant distance geometric information around the neighborhood at each point and obtain all matched umbrellas incident to a input point in the UFM algorithm.

However, the three-level priority queuing mechanism is unable to identify relevant sharp features. It fails to reconstruct sharp features in output triangle mesh because the method only involves distance information among input points and geometric information based on the matching results in the neighborhood for each point. In order to reconstruct triangle mesh with the sharp features in a unified meshing process, it is necessary to introduce a new priority queuing level based on the dependable feature sensitive information in the neighborhood at each input point.

5.4.2. Improved Priority Queuing for Sharp-feature Preservation

Since sharp features could be positively identified through intersections of relative flat local patches around its neighborhood, the proposed algorithm attempts to shape relevant sharp features in the reconstructed triangle mesh by building reliable local umbrella meshes out of “good” flatness in the adjacent region. Therefore, a novel flatness sensitive filter, referred to as the normal vector cone (NVC) filter, is introduced and designed to seek the reliable umbrella with “good” flatness in the neighborhood of relevant sharp features. The successful building of an umbrella with “good” flatness in and around the desired sharp features will help preserve them well in the reconstructed triangle mesh.

Since the estimated matched umbrellas based on M_f or $M_{f(v)}$ are in fact a kind of refined local mesh neighbors, useful geometric information in the neighborhood of each point can be extracted. Additionally, experience from umbrella matching exercise indicates that umbrellas located in the non-smooth or the sharp regions seldom match each other fully by the size information of their Delaunay triangles only. This implies that the NVC filter can be effectively employed into most points located in the sharp feature regions or its adjacent region. With the help from a global estimation based on the dihedral angle of any two matched umbrella facets, the introduced NVC filter make it possible to successfully build an umbrella with “good” flatness close to the desired sharp features.

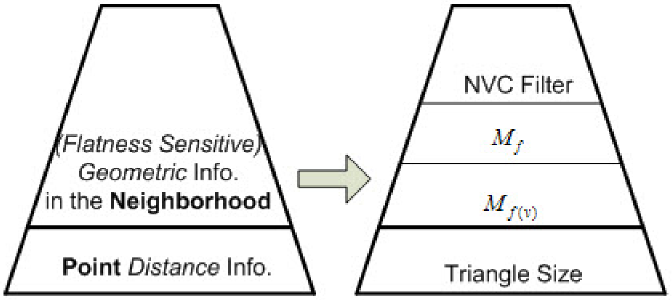


Figure 5.5 Feature sensitive priority queuing

In Figure 5.5, the NVC filter level is added as the new top level of priority queuing. This improved four-level inheritance priority queuing addresses the flatness sensitive geometric information in the neighborhood of each point and is critical for reliable reconstruction of sharp features. The introduced NVC filter will gradually nudge any partially matched umbrella and shape them into fully matched umbrella with “good” flatness in relevant region, through the geometric criteria of normal vector deviation based on the dihedral angle of the matched umbrella facets. It is a unified and progressive triangle mesh reconstruction process. Once the fully matched umbrella for every input point is found, the algorithm finally converges and a watertight manifold triangle mesh is constructed with well-preserved sharp features.

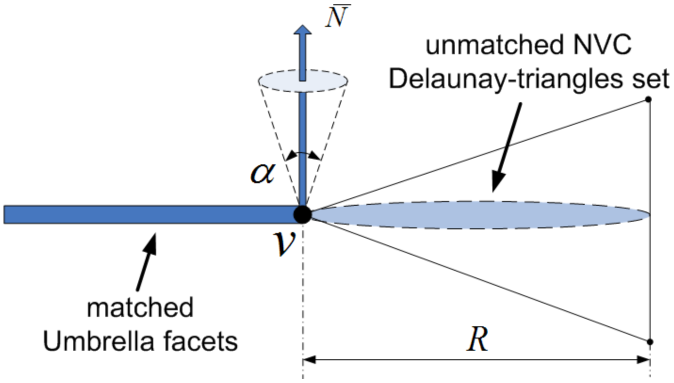


Figure 5.6 Normal vector cone at point v

5.4.3. Normal Vector Cone Filtering

As a novel geometric heuristics filter, the normal vector cone (NVC) filter is designed to reliably reconstruct the sharp features of the output triangle mesh in current work. As shown in Figure 5.6, the NVC filter at point v is defined by three parameters: the cone angle α, the nominal normal $\bar{N}$ and the limitation range R. The nominal normal $\bar{N}$ and the cone angle α could define a normal vector cone at point v, as illustrated by the dashed cone in Figure 5.6. Once the acute angle between the normal of any unmatched Delaunay triangle and the nominal normal $\bar{N}$ at point v is less than the cone angle α, the unmatched Delaunay triangle at v is chosen as a likely candidate for unmatched NVC Delaunay-triangles set. If the size of the preliminary candidate triangle can be further constrained into the limitation range R, it is then qualified to become a member of the final unmatched NVC Delaunay-triangles set. We can also say that if the unmatched Delaunay triangle at point v locates inside its normal vector cone then it is an unmatched NVC Delaunay-triangle. The cone angle α and the nominal normal $\bar{N}$ constrain the orientation of a triangle candidate (an unmatched Delaunay triangle) at point v and the limitation range R limits its size. More details on these three parameters of the NVC filter are described later.

In current work, there exist two types of NVC filters depending on different choice of the cone angle α and the calculation of the nominal normal $\bar{N}$. The first is a generic filter with a larger cone angle ($\alpha = \pi / 4$) and the second is the flatness sensitive filter with a smaller cone angle depending on the global flatness estimation of reconstructed triangle mesh. The calculation of the nominal normal $\bar{N}$ for these two NVC filters are all based on the normal vector estimation algorithm proposed in chapter 4, with an additional weight value being assigned to calculate the $\bar{N}$ for the flatness sensitive filter. Those specific points are identified to apply the flatness sensitive NVC filtering by analyzing their local neighborhood meshes, whose umbrellas likely keep the "good" flatness. The flatness sensitive NVC filtering mainly devotes to reconstructing the sharp features, while the generic NVC filter can help the algorithm converge on finding all fully matched umbrellas.

Cone angle

The value of the cone angle α for the flatness sensitive NVC filter is derived from the global flatness estimation of all matched umbrella facets. For triangle meshes, the dihedral angle defined by the normal of two adjacent triangles is often regarded as the flatness estimation or sharp features indicator. The application of the dihedral angle is based on the idea of normal vector deviation and conceptually straightforward. The dihedral angle can be formulated by the following Equation 5.1:

$$\theta_{dihedral} = \cos^{-1}\left(\frac{\bar{N}_i}{|\bar{N}_i|} \cdot \frac{\bar{N}_j}{|\bar{N}_j|}\right) \qquad \textbf{(5.1)}$$

The variable $\bar{N}_i$ and $\bar{N}_j$ correspond to the normal of the two adjacent triangles.

Generally, a threshold is assigned to the dihedral angle to identify the flat or sharp features. However, the determination of the threshold is always a non-trivial and often challenging task. Although in theory a locally self-adaptive threshold should be preferred over a fixed global threshold, a fixed global threshold is used in current algorithm for the sake of simplicity and efficiency. Doubling the average of the dihedral angles for all matched umbrella facets is found to be a good overall threshold angle between the physical objects and the ideal simulated models. The overall threshold derived from the global estimation of dihedral angles in matched umbrella facets is assigned to the cone angle α. For the generic NVC filter at each point, the value of the cone angle α is often assigned to $\pi / 4$. This kind of coarse angle constrain help drive the umbrellas matching each other due to the consistence of the normal vector estimated by algorithm proposed in chapter 4.

Nominal normal

The calculation of the nominal normal $\bar{N}$ at each input point is based on normal vector estimation algorithm proposed in chapter 4, which depends on the estimated matching results of its umbrella. For certain specific points with the flatness sensitive NVC filter, additional weight values are assigned to calculate their nominal normal $\bar{N}$. These points with the flatness sensitive NVC filter are identified first by analyzing their local neighborhood meshes. The identification process is detailed in the following section.

Identification of point with a potential flat umbrella

As shown in Figure 5.2, an umbrella at v includes a center vertex v, a triangular facets set $U(f)_v = \{f_1, \ldots, f_n\}$ and a circumjacent neighborhood vertex set $U(p)_v = \{p_1, \ldots, p_n\}$. There are three different matching stages for an umbrella in the umbrella matching results, as illustrated in Figure 5.3. If there exists a fully matched umbrella at point v, it is marked by $G_v = 1$. Otherwise, the point v is marked by $G_v = 0$. Consequently, four different types of point in current algorithm can be defined in Table 5.1.

Table 5.1 Classification of point with an umbrella

Name	Grade of Vertex	Shape	Definition
Void Matched Umbrella Point ($\widetilde{U}_v$ Point)	$G_v = 0$	Small Empty Dot	If $\forall f \in U(f)_v : M_f < 3$
Partially Matched Umbrella Point ($\bar{U}'_v$ Point)	$G_v = 0$	Large Empty Dot	If $\exists f \in U(f)_v : M_f = 3$, and $\exists f \in U(f)_v : M_f < 3$
Fully Matched Umbrella Point ($\bar{U}_v$ Point)	$G_v = 1$	Large Solid Dot	If $\forall f \in U(f)_v : M_f = 3$, and $\exists p \in U(p)_v : G_p = 0$
Finished Fully Matched Umbrella Point (Finished $\bar{U}_v$ Point)	$G_v = 1$	Small Solid Dot	If $\forall f \in U(f)_v : M_f = 3$, and $\forall p \in U(p)_v : G_p = 1$

If neither facet of an umbrella at point v is the matched umbrella facet (if $\forall f \in U(f)_v : M_f < 3$), the point v is named the void matched umbrella point ($\widetilde{U}_v$ point), shown by small empty dots in Figure 5.7. The partially matched umbrella points ($\bar{U}'_v$ point) are the ones with an umbrella where triangular facets are only partially matched (if $\exists f \in U(f)_v : M_f = 3$ and $\exists f \in U(f)_v : M_f < 3$). They are shown by large empty dots in Figure 5.7. Once there exists a fully matched umbrella $\bar{U}_v$ at point v with all its circumjacent neighborhood points being non-empty dots (if $\forall f \in U(f)_v : M_f = 3$ and $\forall p \in U(p)_v : G_p = 1$), the point v is named the finished fully matched umbrella point (finished $\bar{U}_v$ point), as illustrated by small solid dots in Figure 5.7. Large solid dots in Figure 5.7 denote points with a fully matched umbrella $\bar{U}_v$ where there exist empty dots in its circumjacent neighborhood vertex set (if $\forall f \in U(f)_v : M_f = 3$ and $\exists p \in U(p)_v : G_p = 0$), which are named as fully matched umbrella point ($\bar{U}_v$ point).

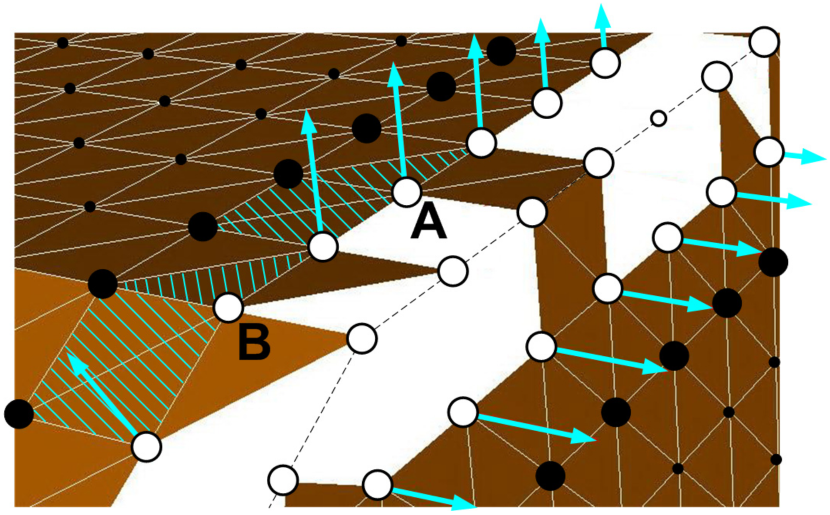


Figure 5.7 Four different point types

In Figure 5.7, only fully matched umbrella facets ($M_f = 3$) are drawn, which are triangles extracted from Delaunay triangles set. The umbrellas of the finished $\bar{U}_v$ points (small solid dots) represent the finished fully matched triangle-mesh region. The umbrellas at the $\bar{U}_v$ points (large solid dots) represent the finished fully matched triangle-mesh region adjacent to the unfinished triangle-mesh region. The partially matched umbrella $\bar{U}'_v$ points (large empty dots) constitute the boundary of the unfinished triangle-mesh region. Those points with a potential flat umbrella will be identified from the partially matched umbrella $\bar{U}'_v$ points.

The partially umbrella $\bar{U}'_v$ points A and B in Figure 5.7 are selected to exemplify how to identify the point with a potential flat umbrella and calculate its nominal normal $\bar{N}$ with additional weight values. For point A and B, all their matched umbrella facets connected with a fully matched umbrella $\bar{U}_v$ point (large solid dot) are picked, referred to the shadow triangles as show in Figure 5.7. These shadow umbrella facets cluster shape together a dependable near-neighborhood region for points A and B respectively. Their interconnection with full umbrella $\bar{U}_v$ points (large solid dot) often means the back of these picked shadow umbrella facets is against a finished fully matched triangle-mesh region (shaped by the umbrellas at $\bar{U}_v$ points or finished $\bar{U}_v$ points). The current algorithm employed the same global flatness based on the dihedral angle in cone angle α estimation to identify whose shadow umbrella facets cluster can be regarded as the flat one. In the case of Figure 5.7, while the $\bar{U}'_v$ point A is identified as a point with a potential flat umbrella but the $\bar{U}_v'$ point B is not because one of its dihedral angles among its shadow umbrella facets cluster is larger than the threshold of the estimated global flatness. The shadow triangles cluster at point A will receive the additional weight value for the calculation for nominal normal $\bar{N}$ at point. In the normal vector estimation algorithm of chapter 4, the original weight value in these shadow triangles cluster is simply the matching index ($M_f = 3$) of them. Once point A is identified, the weight value for its shadow triangles will get an increment ($M_f + 1$) and the nominal normal $\bar{N}$ at point A is calculated according to the normal vector estimation algorithm proposed in chapter 4. A flatness sensitive NVC filtering is employed into the identified points to build an updated priority queue for a potential flat umbrella, such as point A. Otherwise, a generic NVC filtering would be applied, such as the case for point B.

All other all identified points in this case study with additional weight estimated nominal normal $\bar{N}$ have been shown in Figure 5.7. The flatness sensitive NVC filtering employed into them makes it possible to build the desired umbrellas with “good” flatness. As mentioned before, from our experience in umbrella matching, umbrellas located in the non-smooth or the sharp regions often neither fully nor quickly match each other (shape the boundary of the unfinished triangle-mesh region around the sharp features) due to the perturbation of the scanned point cloud data that often plagues these regions. In summary, by resorting to the flatness sensitive NVC filtering and through the successful building of an umbrella with “good” flatness adjacent to the sharp features, the proposed algorithm can effectively preserve the desired sharp features in the reconstructed surface. The flowchart of Normal Vector Cone filtering is shown in Figure 5.8.

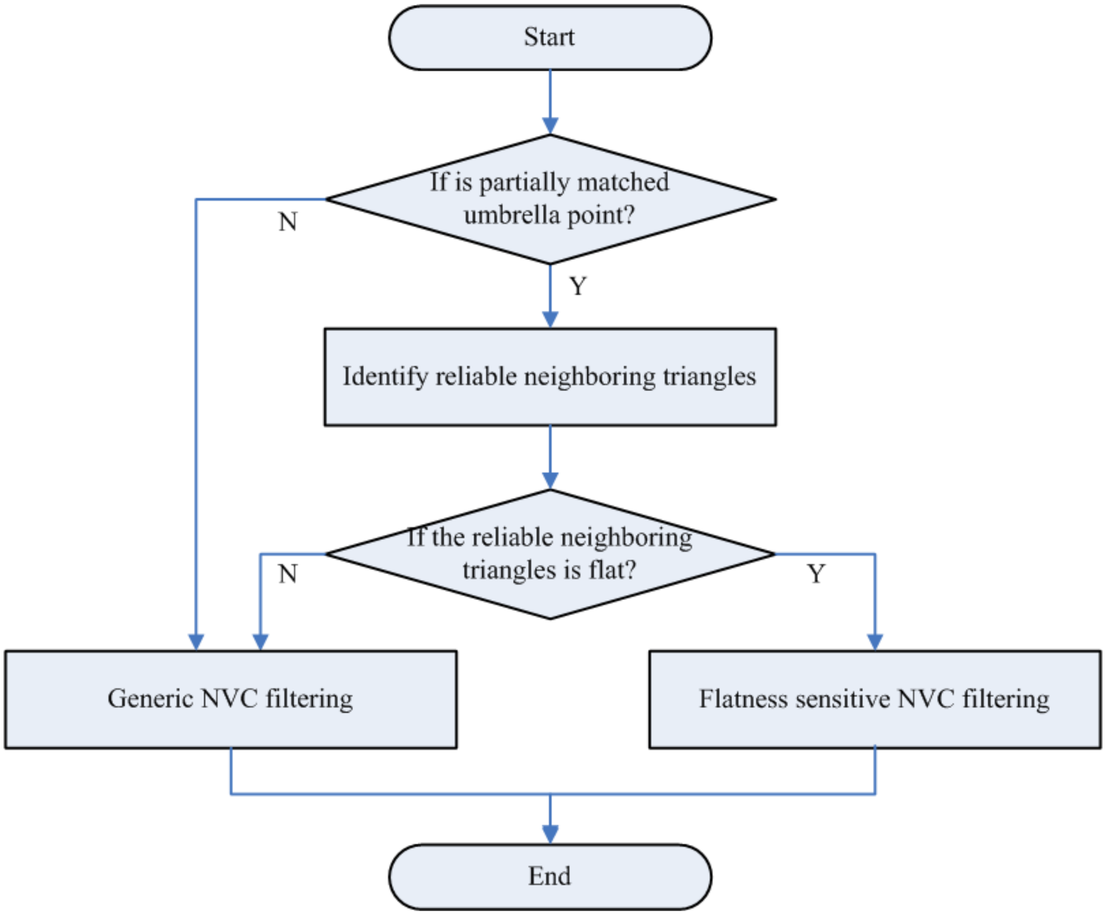

Start
If is partially matched umbrella point?
N
Y
Identify reliable neighboring triangles
If the reliable neighboring triangles is flat?
N
Y
Generic NVC filtering
Flatness sensitive NVC filtering
End

Figure 5.8 NVC filtering flowchart

Limitation range

The limitation range R is designed to further screen selected candidates for unmatched NVC Delaunay-triangles set at each point. For the candidate triangles preliminarily chosen in NVC Delaunay-triangles set, they are further filtered by checking the Limitation Range R. The limitation range R is defined and measured by the concept of the ring of an umbrella. All circumjacent neighborhood vertices $U(p)_v = \{p_1, \dots, p_n\}$ in the umbrella U_v at point v are regarded as an one-ring neighboring points set at point

v. If one-ring neighboring points set of all points in the one-ring neighboring points set ($U(p)_v$) are also included in the neighboring points set for point v (excluding itself), we term it two-ring neighboring points set at point v. If unmatched NVC Delaunay-triangles set is limited to one-ring neighborhood points set, all candidate triangles in unmatched NVC Delaunay-triangles set, i.e. one of whose vertices is not in the one-ring neighboring points set, must be removed from the unmatched NVC Delaunay-triangles set. In current algorithm, two-ring neighboring points set are selected as the value for the limitation range R to further limit the unmatched NVC Delaunay-triangles set, denoted by $R = 2$. It provides a compromised selection for different sampling density in point cloud data. It should be noted that filtering in the unmatched NVC Delaunay-triangles set only means these candidate triangles become that part of final reconstructed umbrellas with higher priority than those unmatched non-NVC Delaunay-triangles set.

The Delaunay triangles set incident to a point is filtered by the NVC filter based on three parameters: the cone angle α, the nominal normal $\bar{N}$ and the limitation range R. The flatness sensitive NVC filtering makes it possible to construct a fully matched local umbrella with "good" flatness, likely adjacent to the desired sharp features. The output of these flat umbrellas located in the neighborhood of the sharp features plays a significant role in shaping the sharp features in the final reconstructed triangle mesh. The advanced four-level inheritance priority queuing mechanism makes the proposed algorithm a feature sensitive triangle mesh reconstruction algorithm via the unified progressive local mesh matching process. The reconstructed triangle mesh promises to preserve sharp features well and pass through all the original input points without adding or removing any points. Multiple case studies have been carried out and analyzed to validate the performance of the proposed algorithm.

5.5. Implementation and Results

The quality of a reconstructed surface varies widely and is often determined by sampling condition of input points, noise content of measured points, as well as specific shape of the original object. For the proposed algorithm based on Delaunay triangulation, it is assumed that the points are noise-free and other geometric information, such as the normal direction at each point, is unavailable. The measured point cloud is also unorganized and the original object surface needs not be smooth. Finally, there is no limitation on the genus of the original object surface.

Algorithm implementation

The proposed algorithm is programmed in C++ language. Codes in the Computational Geometry Algorithms Library CGAL [43] are applied to complete the 3D Delaunay triangulation (DT) of targeted data set. Also, for topological information, another open source C++ template library, VCG Library [47], is employed for storing and processing of the triangular and tetrahedral meshes. The experimental case studies are performed on a Windows-based PC with a 2.66GHz processor and 4GB memory.

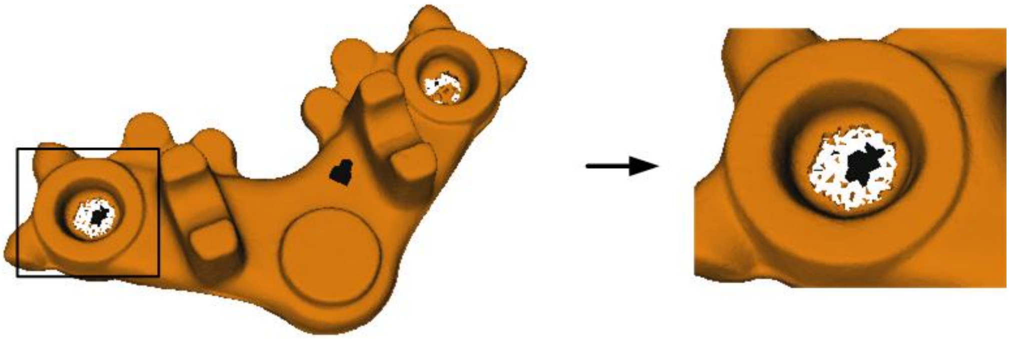
Table 5.2 Implementation results for some public point cloud data sets

Data Set			**Implementation Results**			
Name	Genus	Input (points)	Output (triangles)	Matching	ΔF	Time (sec.)
Bunny	0	35,947	71,890	100%	0	72.44
Golf Club	0	16,585	33,166	100%	0	29.18
Mechpart	3	4,102	8,212	100%	0	9.47
3Holes	3	4,000	8,008	100%	0	6.15
Knot	1	10,000	20,000	100%	0	17.83
Mannequin	0	12,772	25,540	100%	0	35.76
Casting Die	0	63,613	127,230	100%	8	141.71
Oilpmp*	0	30,937	61,862	100%	0	77.55
Rocker Arm	1	10,044	20,088	100%	0	23.47
Screwdriver	0	27,152	54,300	100%	0	61.06
Hand	0	25,001	49,998	100%	0	49.93
Teapot	1	25,667	51,334	100%	0	86.70
Golf Head	0	52,524	105,044	100%	0	126.90
Foot	0	20,021	40,038	100%	0	36.60
Fandisk01	0	6,475	12,946	100%	0	9.17
Fandisk02	0	16,475	32,946	100%	0	47.16
SimulationSolid	0	6,988	13,972	100%	0	7.71
CubewithHole	1	2,224	4,448	100%	0	3.88

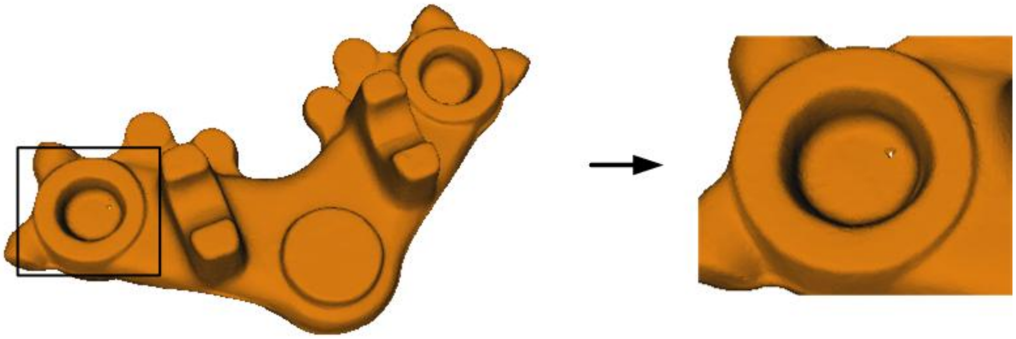
* *Oilpmp included 4 repeated points and they were removed.*

Table 5.2 shows the computational efficiency and effectiveness for a number of publicly available point cloud data sets. It can be seen that 100% matching has been attained for all the test cases with different genus. The matching percentage represents the ratio of the number of the resulting points with fully matched umbrellas to the number of points in the complete input point cloud. A matching ratio of 100% indicates that fully matched umbrellas at all points are found and a watertight manifold triangle mesh is reconstructed successfully. ΔF is a parameter designed for estimating the topological quality of the reconstructed surface compared to the original model, as described in Equation 3.2 of chapter 3. According to Equation 3.2, lower ΔF values indicate smaller topological difference and zero ΔF represents homeomorphism between the reconstructed triangle-mesh surface and the original model surface. Our results from current extended UFM algorithm in Table 5.2 have shown the best topological quality so far compared to Table 3.3 and Table 3.4 in chapter 3, although keep in mind that there is no theoretical guarantee for topological quality.

As shown in Figure 5.9b, the reconstructed surface for Casting Die model in current algorithm is a watertight manifold triangle mesh and interpolating all points in a point cloud, though it contains a minor topological deviation ($\Delta F = 8$). Compared to the result from the UFM algorithm proposed in chapter 3 shown in Figure 5.9a, there is an apparent improvement on the output topological quality and algorithm convergency for the current extended UFM algorithm. All resulting 3D rendering images of the reconstructed triangle-mesh surface for these testing points cloud data sets are shown in Figure 5.10.

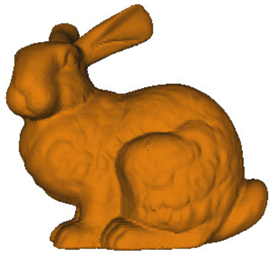
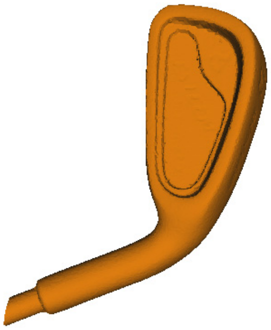
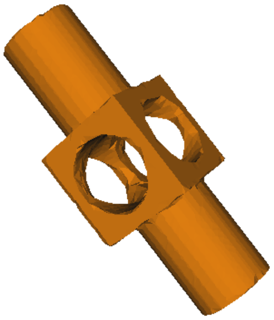


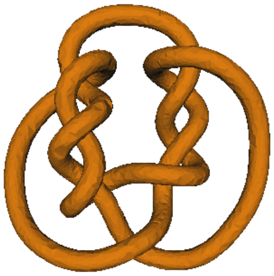
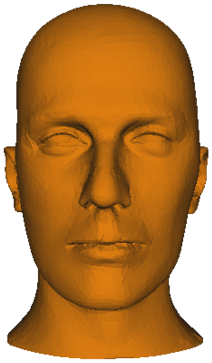
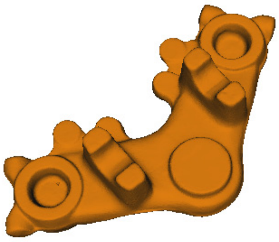
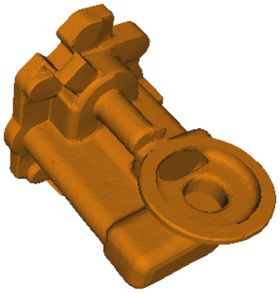
(a) UFM algorithm in chapter 3

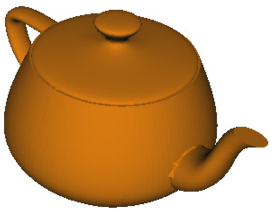


(b) current extended UFM algorithm

Figure 5.9 Comparison of reconstructed surface for Casting Die data set



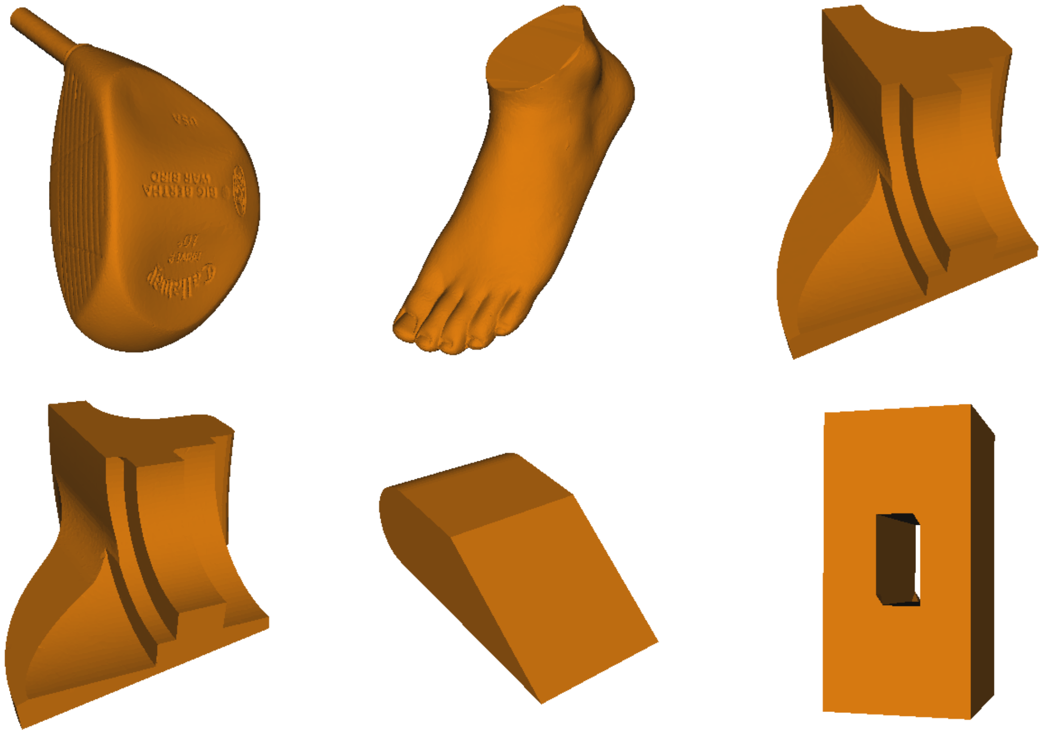


Figure 5.10 Reconstructed surface for the data set in Table 5.2

Results analysis

For comparison study, three existing algorithms are selected: the UFM algorithm (presented in chapter 3) based on the authors' previous work [64], two other well-known algorithms, namely, the cocone algorithm and the tight cocone algorithm, developed by Amenta et al. [40] and Dey et al. [2] respectively. All three algorithms are selected as general smooth triangle mesh reconstruction algorithms. The cocone algorithm is not designed for watertight surface reconstruction. The UFM and tight cocone algorithm are both targeting for watertight surface reconstruction. The binary codes of the cocone and tight cocone algorithms are readily available on the Internet [84].

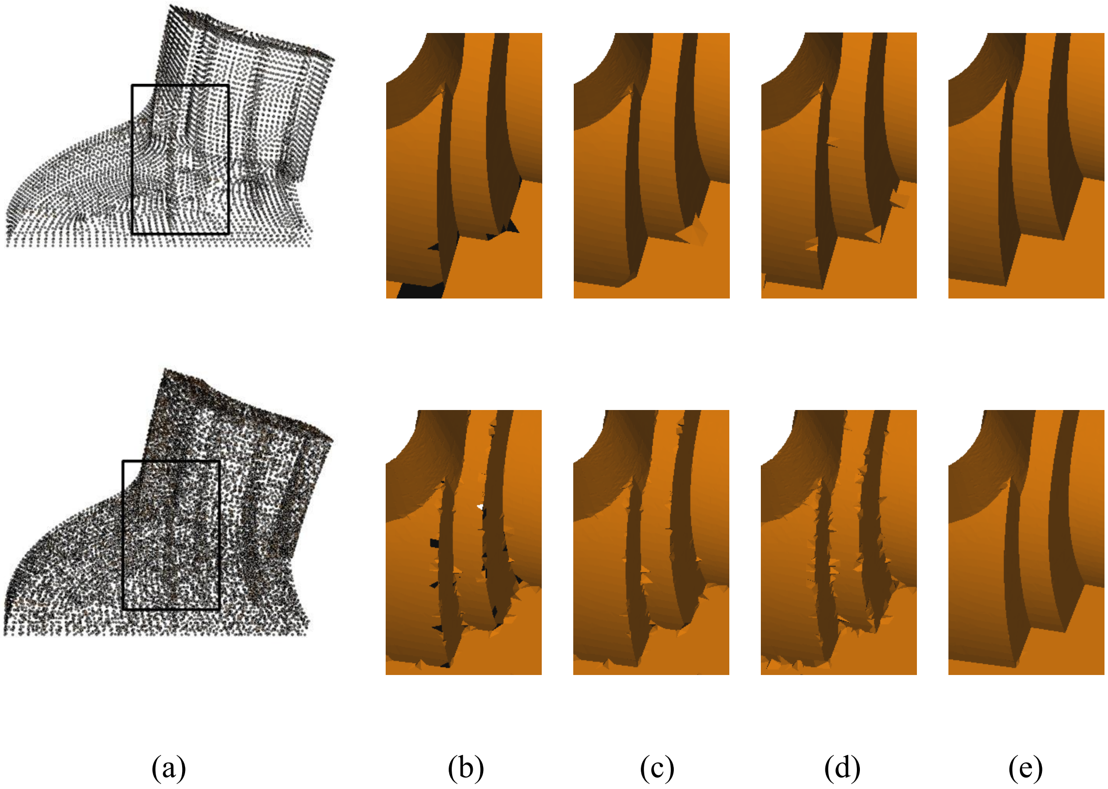
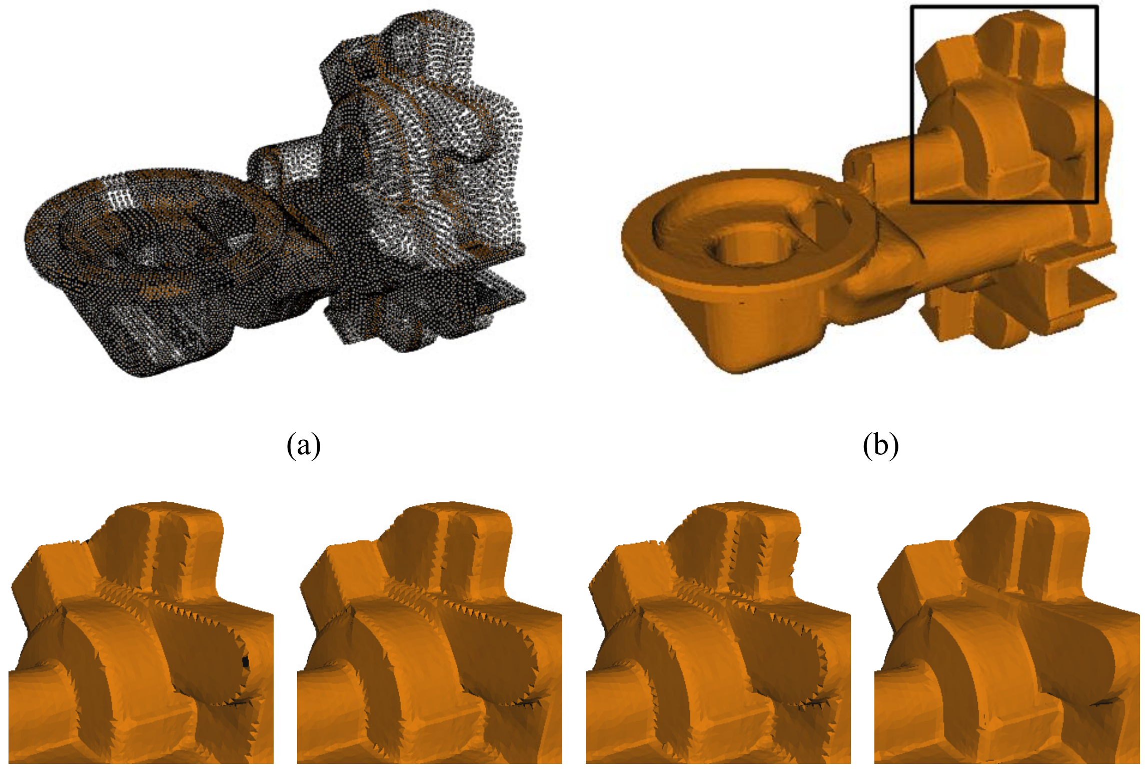


Figure 5.11 Comparison of resulting meshes of Fandisk01 and Fandisk02 data set

Figure 5.11 demonstrates comparison results for two well-known Fandisk models with sharp features. The point cloud data of Fandisk01 model is a typical point set with uniform distribution and shown in the first row. The point cloud data of Fandisk02 model is a typical point set with non-uniform distribution and shown in the second row. Figure 5.11a is the raw point clouds data for these two models. Chosen region for algorithm comparison is marked out with a rectangle. Figure 5.11b and Figure 5.11c are results from the cocone and tight cocone algorithm, respectively. Figure 5.11d is the result from the UFM algorithm of chapter 3. The last column, Figure 5.11e, is the output of the proposed extended UFM algorithm in this chapter. Cross comparison among the output of these algorithms clearly shows that the proposed extended UFM algorithm stands out and reconstruct perfectly the original model with its sharp features. The point clouds data of both Fandisk models are typical examples of feature sensitive sampling.

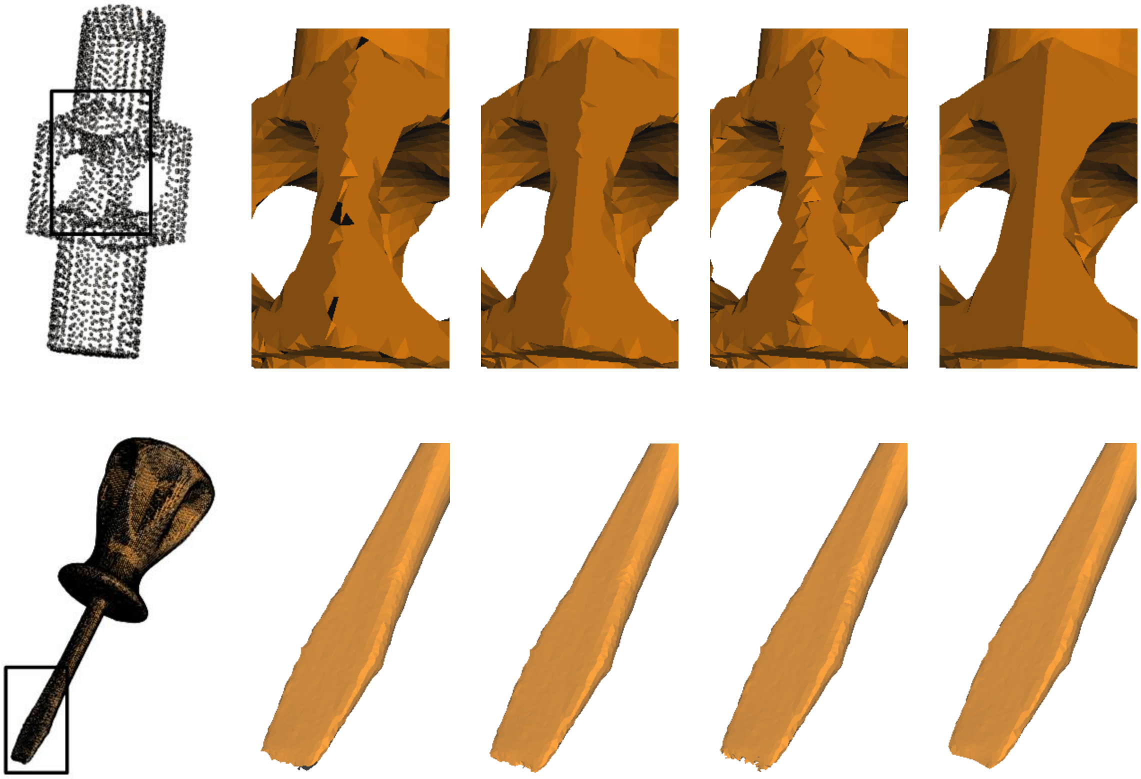


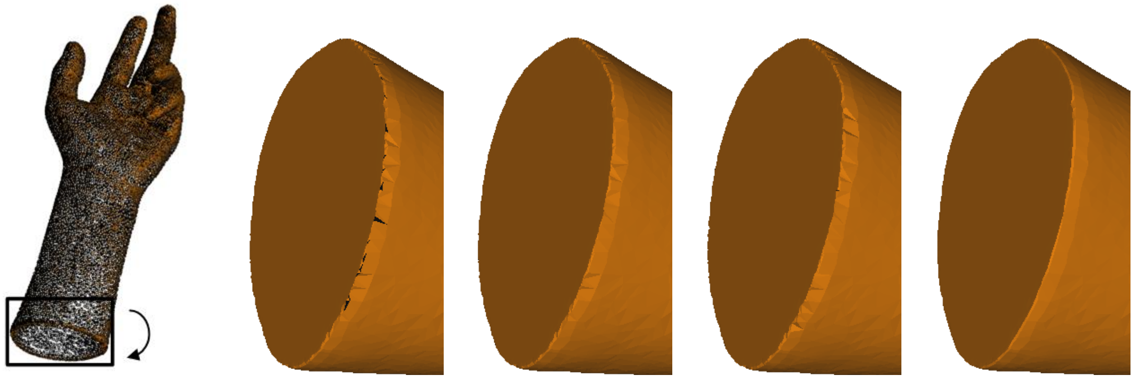
(a) (b)

(c) (d) (e) (f)

Figure 5.12 Comparison of resulting meshes of Oilpmp data set

Another comparison comes from the Oilpmp model, as shown in Figure 5.12. It includes lots of features typical of mechanical parts and possesses non-uniform sampling. Figure 5.12a is the raw point cloud of Oilpmp model loaded with sharp features and Figure 5.12b indicates highlighted region in Oilpmp model for detailed comparison. Figure 5.12c and Figure 5.12d are output triangle meshes from the cocone and tight cocone algorithms. Figure 5.12e is the result from the UFM algorithm presented in chapter 3. The output of the proposed algorithm in this chapter is shown in Figure 5.12f. Evidently, the output of the current algorithm out performs others by preserving all the sharp features and generating a watertight manifold triangle-mesh surface. More comparison examples from realistic scanned point cloud data are show in Figure 5.13. The reconstructed surfaces in Figure 5.13b, Figure 5.13c, Figure 5.13d and Figure 5.13e comes from the cocone algorithm, tightcocone algorithm, UFM algorithm in chapter 3 and the current extended UFM algorithm respectively. The extended UFM algorithm proposed in current chapter shows the best overall performance for the recovery of the sharp features.





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

Figure 5.13 Comparison of resulting meshes with sharp features of some examples

Additional comparisons for the feature sensitive sampling in ideal simulation data set are performed. Two simulated models, SimulationSolid and CubewithHole, are selected for this testing. The presented feature sensitive algorithm demonstrates a more consistent reconstructed triangle mesh with sharp features, compared with the other three algorithms. The comparison results for the two simulated models are shown in Figure 5.14. The cocone, tight cocone and UFM (proposed in chapter 3) algorithms are shown in Figure 5.14a, Figure 5.14b and Figure 5.14c respectively. The last panel shown in Figure 5.14d is the output of the current extended UFM algorithm presented in this chapter. As true with all simulation data set, the original triangle-mesh surfaces of these two models are known and their geometric properties can also be estimated. For better comparison of the shape deviation of reconstructed surface from different algorithms, the normal deviations between the reconstructed surfaces and the original model are calculated. Their colour maps are shown in Figure 5.15 with the same ordering sequence of Figure 5.14. The region with green colour represents a larger shape deviation than the region with blue colour. Evidently the best surface reconstruction with sharp features goes to the last column in Figure 5.15d, which is the output of extended UFM algorithm proposed in this chapter. Almost all sharp features can be correctly reconstructed including the sharp feature with the acute angle in SimulatinSolid model.

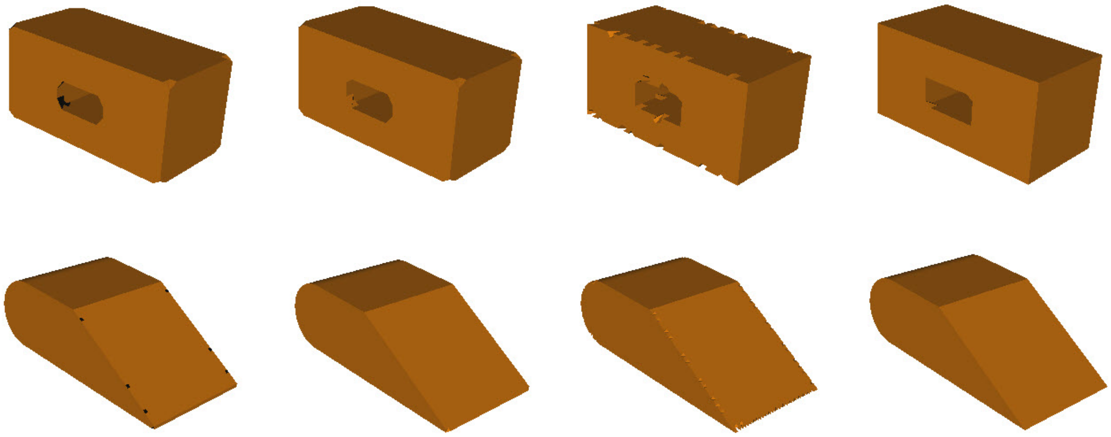

(a) (b) (c) (d)

Figure 5.14 Shape comparison of resulting meshes of simulated data set

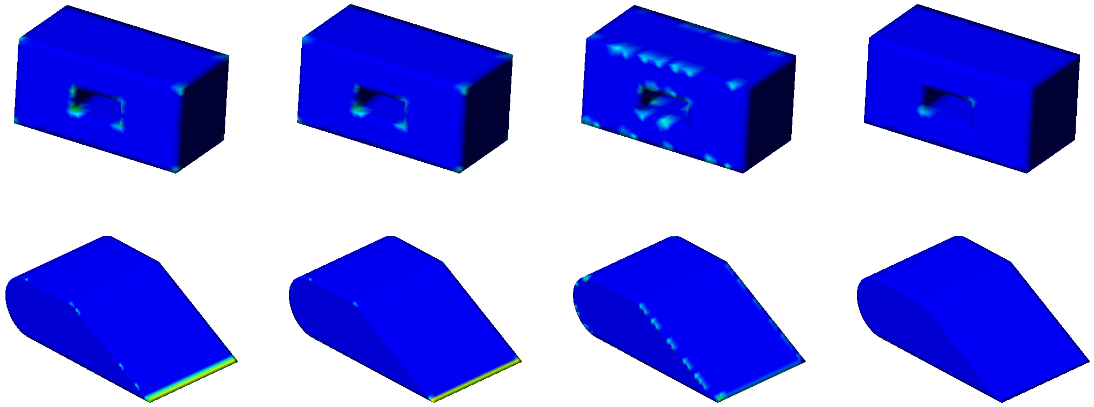

(a) (b) (c) (d)

Figure 5.15 Colour map comparison of resulting meshes of simulated data set

Limitations

Although the presented feature sensitive algorithm improves the quality of the reconstructed surface with sharp features, it may still miss some sharp features, as shown in Figure 5.16. The distribution of the input point cloud of the Mechpart model is highly non-uniform and many features can not be reconstructed in the absence of sampling density.

Furthermore, we cannot guarantee that the proposed algorithm does not miss any feature under arbitrary sampling density condition. How to define a sufficient sampling condition for non-smooth geometry is still an open question in surface reconstruction. The presented algorithm uses the flatness threshold based on the global dihedral angle estimation in reconstructed triangle mesh to determine whether an umbrella is a local patch with "good" flatness. A self-adaptive local threshold can possibly perform better in identifying relevant sharp features. The geometric error evaluation in different models is also beyond the scope of this chapter. All these outstanding issues will be considered and addressed in our future work.

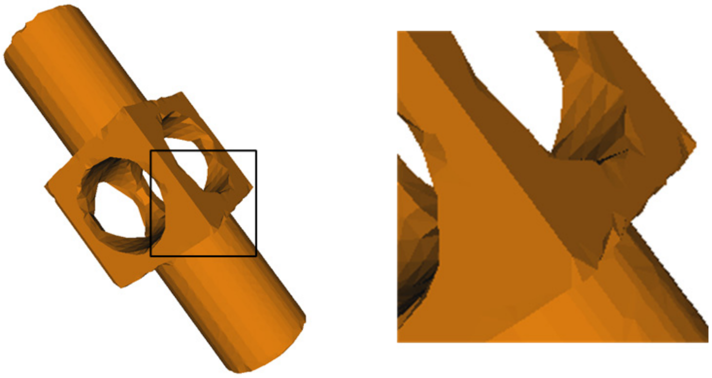


Figure 5.16 Loss of sharp features

5.6. Concluding Remarks

A novel feature sensitive mesh reconstruction method is presented in this chapter. It is based on dependable geometric information in the neighborhood of each input point. The dependable geometric information is derived from the matching results of the local umbrella mesh constructed at each input point. The core idea of the proposed algorithm is to seek reliable local umbrella meshes with good flatness in the adjacent region to help shape the relevant sharp features in reconstructed triangle mesh. A new flatness sensitive filter, referred to as the normal vector cone (NVC) filter, is introduced to seek for the reliable adjacent umbrella with good flatness in the neighborhood of relevant sharp features. Depending on a unified multi-level priority queuing mechanism, the presented algorithm can automatically and reliably reconstruct the watertight manifold triangle mesh with sharp features in an integrated reconstruction process without any post-processing need. It should be noted that there exist two types of NVC filters with different cone angles. The first is a generic filter with a larger cone angle ($\pi / 4$) and the second is the flatness sensitive filter with a smaller cone angle depending on the global flatness estimation of reconstructed triangle mesh. The flatness sensitive NVC filter mainly focuses on reconstructing the sharp features and the generic NVC filter can help the algorithm converge on finding all fully matched umbrellas. The experimental results have shown that the proposed algorithm can improve the reconstructed triangle mesh quality and reduce the shape deviation compared to the original model geometry.

As discussed in previous section, the presented algorithm may still miss some sharp features in the reconstructed triangle mesh for highly non-uniform or under-sampled region of the input point cloud data. In the future, replacing the current fixed global flatness threshold estimation with the local self-adaptive estimation is expected to further improve the shape deviation of the reconstructed surface and better preserve sharp features. Additionally, a post-processing algorithm for the complete recovery of the sharp features is an interesting research direction in the future, such as the remeshing processing based on moving or adding some reference points.

6 CONCLUSIONS AND FUTURE WORK

This thesis is devoted to watertight manifold triangle mesh surface reconstruction with emphasis on the recovery of the sharp features. The reconstructed triangle mesh surface interpolates (passes through) all measurement points in an unorganized point cloud data with low-noise. An integrated triangle mesh processing framework for surface reconstruction based on Delaunay triangulation is presented in the thesis. The proposed main algorithm, Umbrella Facet Matching (UFM) algorithm, features a unified multi-level inheritance priority queuing mechanism for seeking and updating the optimum local manifold mesh at each data point. Its two extended algorithms are then presented to further improve the quality of reconstructed triangle mesh surface. Both algorithms resort to the same multi-level inheritance priority queuing mechanism to analyze local neighbourhood mesh at each data point. Through the integrated surface reconstruction framework and the extended geometric heuristics proposed in the thesis, the resulting reconstructed surface can effectively recover the sharp features in the original physical object and capture their topology and geometric shapes reliably. The effectiveness of these algorithms has been demonstrated using both simulated and real-world point cloud data sets. For each algorithm, multiple case studies are performed and analyzed to validate its performance.

6.1. Main Contributions

The main contributions of the proposed algorithms in this thesis can be summarized as following:

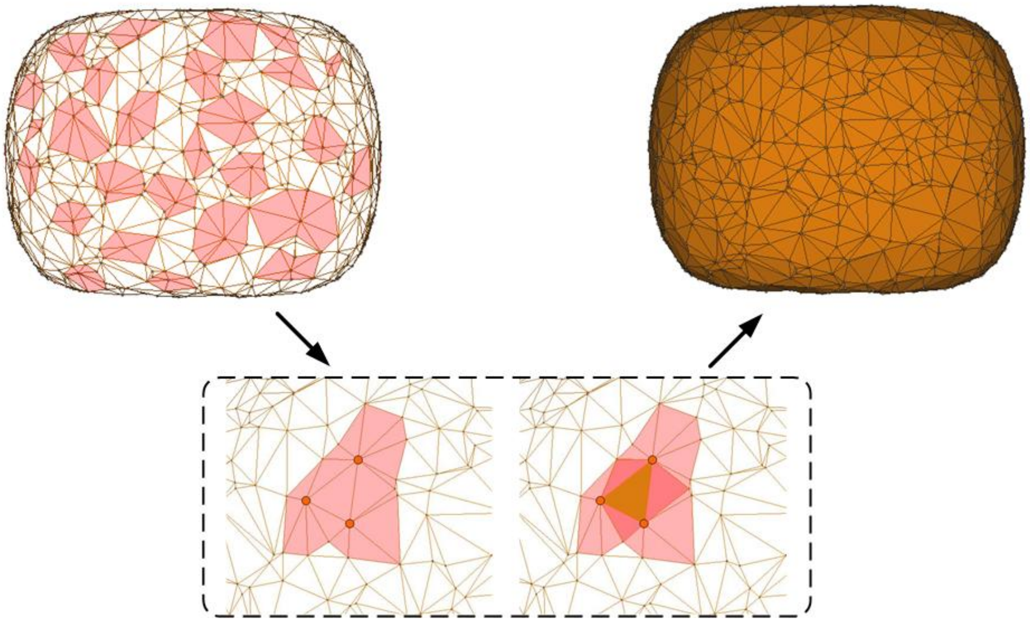


Figure 6.1 **Automatic watertight manifold surface reconstruction via progressive local mesh matching**

Automatic watertight manifold surface reconstruction

This doctoral research proposes an effective approach to automatically reconstruct a watertight manifold triangle-mesh surface interpolating all points in an unorganized point cloud data with low-noise, which is named Umbrella Facet Matching (UFM) algorithm. As illustrated in Figure 6.1, the algorithm starts by forming an initial open umbrella at each point from its Delaunay triangles. If a triangular umbrella facet is included in all of the three umbrellas of its three vertices, the facet is considered a matched facet (the darkest triangles in Figure 6.1). When all the triangular facets of an umbrella are matched facets, the umbrella is defined as a fully matched umbrella. Once the fully matched umbrella for every point is found, a watertight manifold triangle mesh is guaranteed to be constructed (the top right mesh in Figure 6.1).

The multi-level inheritance priority queuing mechanism introduced here (Figure 3.3 in chapter 3) aims to seek the fully matched umbrella at each point by iteration depending on the umbrella facet matching results. A novel evaluation methodology of local mesh matching (Table 3.1 in chapter 3) has been proposed to represent the umbrella facet matching results for the priority queuing. Since the proposed building process of an umbrella is equivalent to a redundant Delaunay triangles removal process depending on a priority queue, an updated priority queue will lead to an updated umbrella. Therefore, the desired fully matched umbrella at each point can be found through a progressively updated priority queue according to the umbrella matching result. The basic idea of the progressive local mesh matching is illustrated in Figure 6.2.

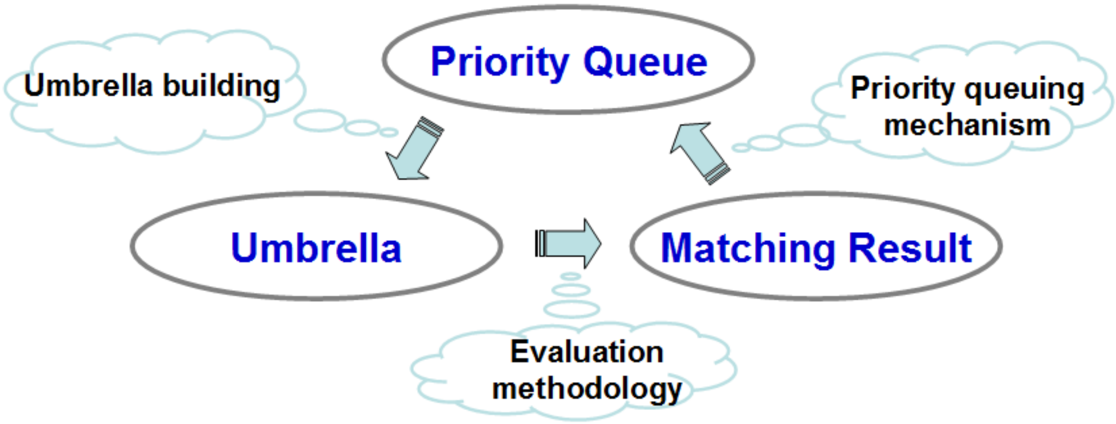
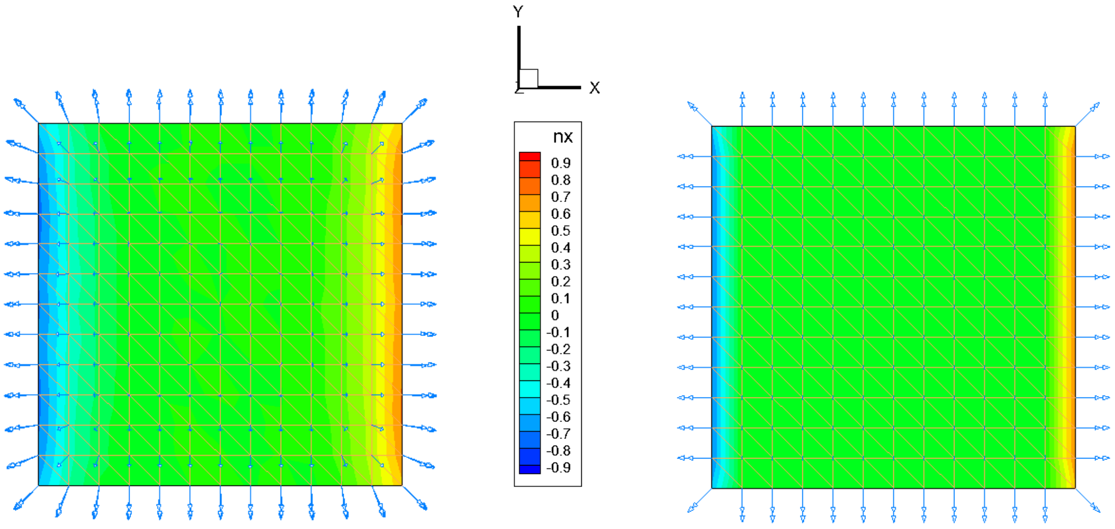


Figure 6.2 Progressive local mesh matching mechanism

Normal estimation based on local mesh matching results

Normal vector is a local geometric property of a 2-dimensional surface and specific to each given point. Therefore, reliable estimation of the normal vector at each point in a point cloud data heavily depends on the positive identification of its valid neighboring points in the neighborhood. A well estimated normal vector is a significant step towards correct reconstruction of sharp features in the original model surface.

The novel evaluation methodology of local mesh matching in this thesis provides a refined way of finding reliable local Delaunay triangulation mesh neighbors at each point in the point cloud data. The well estimated local Delaunay triangulation mesh neighbors at each point become the key towards computing a reliable normal vector at each point, especially for those points adjacent to the sharp features. Comparing with the general numerical optimization approaches, such as least square approach, the proposed combinatorial normal vector estimation algorithm yields more accurate result for low-noise or no-noise point cloud data, though it might be more time-consuming. As shown in Figure 6.3, the normal vectors of simulated point cloud data from a cube model are estimated from both the weighted plane fitting algorithm [59] (Figure 6.3a) and the proposed algorithm in chapter 4 (Figure 6.3b). Figure 6.3a and Figure 6.3b compare color maps of corresponding deviation of estimated normal vector at each point along X axis. At each point its normal vector is marked in red ($nx = 1$) if it points to X axis and in blue ($nx = -1$) if it points to $-X$ direction. For normal vector perpendicular to X axis it is marked in green ($nx = 0$). Evidently, the estimated normal vectors from the proposed algorithm in this thesis demonstrate better results in the region adjacent to the sharp features due to its more accurate local Delaunay triangulation mesh neighbors.

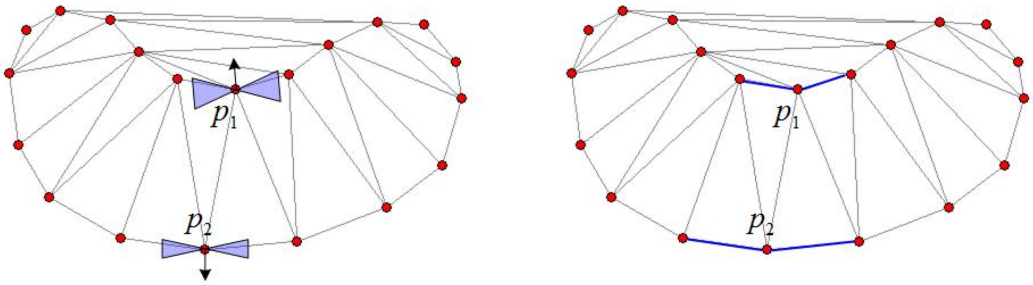


(a) (b)

Figure 6.3 Comparison of estimated normal vector from: (a) weighted plane fitting algorithm; (b) the proposed algorithm in chapter 4

Sharp features reconstruction via normal vector cone (NVC) filtering

Automatic and reliable reconstruction of sharp features remains an open research question in surface reconstruction. The extended UFM algorithm presented in chapter 5 addresses the sharp feature preservation issue in surface reconstruction by analyzing dependable neighborhood geometric information for each input point. Such information is derived from the matching result of the local umbrella mesh constructed at each point. Resorting to the unified multi-level inheritance priority queuing mechanism proposed in chapter 3, a novel flatness sensitive filter, referred to as the normal vector cone (NVC) filter, is introduced and demonstrated to be able to reliably reconstruct sharp features. Figure 6.4 illustrates the basic function of the NVC filter in a curve reconstruction example in Euclidean space R^2. The NVC filters at point p_1 and p_2 can be estimated by analyzing their local mesh matching results respectively (shown in Figure 6.4a). For each point, the candidate segments (Delaunay triangles in Euclidean space R^3) included in NVC filter will be identified and become a part of final reconstructed umbrellas with high priority (blue segments in Figure 6.4b). The NVC filtering is designed to extract neighborhood geometric information reliably and drive the priority queue of umbrella-building at each point. It should be noted that there exist two types of NVC filters with different cone angles. The NVC filter at p_1 is a generic filter with a larger cone angle ($\pi / 4$) and the one at p_2 is a flatness sensitive filter with a smaller cone angle depending on the global flatness estimation of reconstructed triangle mesh. The sharp feature recovery depends more on the flatness sensitive NVC filters, though the generic NVC filters can help the algorithm converge on finding all fully matched umbrellas. Refer to chapter 5 for more details on the NVC filters.



(a) (b)

Figure 6.4 Normal vector cone filtering

With the proposed algorithms, a watertight manifold triangle-mesh surface can be successfully reconstructed, which interpolates (passes through) the complete original point cloud data without point addition or removal. The output surfaces preserve the sharp features well and in most cases contain only minor shape deviations, comparing to the original surface of physical object.

6.2. Future Work

As stated previously, the surface reconstruction research presented in this thesis assumes a low-noise, unorganized input point cloud P in Euclidean space R^3. As a Delaunay-based approach, the proposed algorithms are still sensitive to noise in the input point could data, yet they are demonstrated to work well not only in the low-noise point cloud data, but also guarantee to output a manifold interpolation surface (with few small holes) in very noisy data (shown in Figure 3.14 of chapter 3). Reducing measurement noise in a scanned point cloud data in a separate pre-processing step has become a very active research subject lately, and is being investigated in our research group with ultra-precise 3D laser scanning system. The UFM algorithm and its extended algorithms set presented in the thesis will certainly benefit from these advanced research studies.

Additionally, regions rich with small features, such as high curvatures, usually are not scanned well by laser scanner and often generate either non-uniform or under-sampled point cloud data, especially when these small features are sharp. These sharp features pose great challenging for all approaches on surface reconstruction. Figure 6.5 shows an example of the reconstructed interpolation triangle-mesh surface with sharp features from the proposed algorithms, with some sharp features missing due to the highly non-uniform and under-sampled measurement points around sharp features region. However, our algorithms still demonstrate a great robustness and improved accuracy over other watertight interpolation surface reconstruction algorithms, regardless of the fact that not all sharp features are guaranteed in reconstruction, as illustrated in Figure 5.10 to Figure 5.15 in chapter 5 and Table 5.2 in chapter 5. Building upon the current set of algorithms proposed in this thesis, a complete recovery of the sharp features for computer-aided design and inspection may become possible in the future. One promised research direction is to extend the surface adjacent to the sharp features and calculate the proper intersection position under the specific geometric error. Then the desired complete recovery of sharp features could be accomplished through moving relevant measurement points.

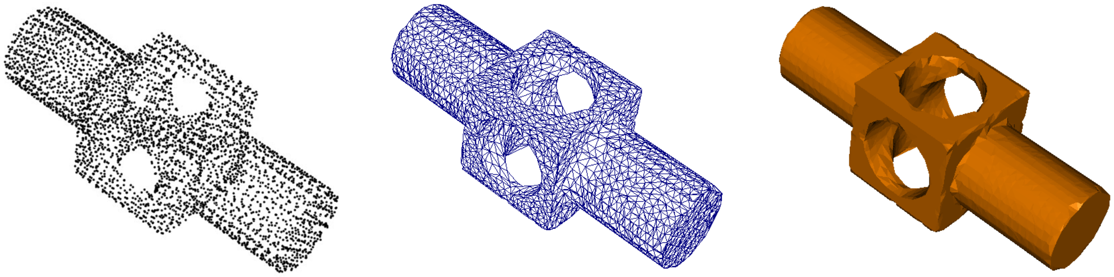


Figure 6.5 An example of the sharp features missed

Most Delaunay-based algorithms begin with computing the entire Delaunay triangulation of the input points cloud data and end with generating an interpolation triangle mesh surface. Although robust and efficient algorithms exist in computing the Delaunay triangulation in Euclidean space R^3 [43], the computation remains time-consuming for massive point cloud data of which it is not uncommon to see tens of millions of points currently in practice. Most of the triangulation result is discarded in the end in these Delaunay-based algorithms. In the future, a profitable research direction is to sort out and compute only the necessary part of the Delaunay triangulation, which would be much desirable and drastically improve the efficiency of Delaunay-based surface reconstruction approaches.

REFERENCES

[1] M. Botsch, M. Pauly, L. Kobbelt *et al.*, "Geometric Modeling Based on Polygonal Meshes," in ACM SIGGRAPH 2007 courses, San Diego, California, 2007, pp. 1.

[2] T. K. Dey, and S. Goswami, "Tight Cocone: A Water-tight Surface Reconstructor," in Proceedings of the 8th ACM Symposium on Solid Modeling and Applications, Seattle, Washington, USA, 2003, pp. 127-134.

[3] F. Cazals, and J. Giesen, *Delaunay Triangulation Based Surface Reconstruction: Ideas and Algorithms*, INRIA, 2004.

[4] T. K. Dey, *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*, New York: Cambridge University Press, 2007.

[5] H. Edelsbrunner, M. J. Ablowitz, S. H. Davis *et al.*, *Geometry and Topology for Mesh Generation*, New York: Cambridge University Press 2006.

[6] J. R. Munkres, *Topology, a First Course*, Englewood Cliffs, NJ: Prentice-Hall, 1975.

[7] J. R. Weeks, *The Shape Space*, New York: Marcel Dekker, 1985.

[8] M. Berg, M. Kreveld, M. Overmars *et al.*, *Computational Geometry: Algorithms and Applications*, New York: Springer, 2000.

[9] C.-C. Kuo, and H.-T. Yau, "A Delaunay-Based Region-Growing Approach to Surface Reconstruction from Unorganized Points," *Computer-Aided Design,* vol. 37, no. 8, pp. 825-835, 2005.

[10] M. Gross, and H. Pfister, *Point-Based Graphics*: Morgan Kaufmann, 2007.

[11] H. Hoppe, T. DeRose, T. Duchamp *et al.*, "Surface Reconstruction from Unorganized Points," in Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, 1992, pp. 71-78.

[12] B. Curless, and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, 1996, pp. 303-312.

[13] G. Turk, and J. F. O'Brien, "Shape Transformation Using Variational Implicit Functions," in Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, 1999, pp. 335-342.

[14] D. Huong Quynh, G. Turk, and G. Slabaugh, "Reconstructing Surfaces by Volumetric Regularization Using Radial Basis Functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 10, pp. 1358-1371, 2002.

[15] J. C. Carr, R. K. Beatson, J. B. Cherrie *et al.*, "Reconstruction and representation of 3D objects with radial basis functions," in Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 2001, pp. 67-76.

[16] Y. Ohtake, A. Belyaev, and H. P. Seidel, "A Multi-scale Approach to 3D Scattered Data Interpolation with Compactly Supported Basis Functions," in Shape Modeling International, 2003, pp. 153-161.

[17] H. Xie, J. Wang, J. Hua *et al.*, "Piecewise C1 Continuous Surface Reconstruction of Noisy Point Clouds via Local Implicit Quadric Regression," in Proceedings of the 14th IEEE Visualization 2003 (VIS'03), 2003, pp. 13.

[18] J.-D. Boissonnat, and F. Cazals, "Smooth Surface Reconstruction via Natural Neighbour Interpolation of Distance Functions," in Proceedings of the sixteenth annual symposium on Computational geometry, Clear Water Bay, Kowloon, Hong Kong, 2000, pp. 223-232.

[19] F. Chazal, and A. Lieutier, "Topology Guaranteeing Manifold Reconstruction Using Distance Function to Noisy Data," in Proceedings of the 22th annual symposium on Computational geometry, Sedona, Arizona, USA, 2006, pp. 112-118.

[20] G. M. Nielson, "Radial Hermite Operators for Scattered Point Cloud Data with Normal Vectors and Applications to Implicitizing Polygon Mesh Surfaces for Generalized CSG Operations and Smoothing," in Proceedings of the conference on Visualization '04, 2004, pp. 203-210.

[21] C. Shen, J. F. O'Brien, and J. R. Shewchuk, "Interpolating and approximating implicit surfaces from polygon soup," *ACM Trans. Graph.,* vol. 23, no. 3, pp. 896-904, 2004.

[22] M. Alexa, J. Behr, D. Cohen-Or *et al.*, "Point set surfaces," in Proceedings of the conference on Visualization '01, San Diego, California, 2001, pp. 21-28.

[23] S. Fleishman, D. Cohen-Or, M. Alexa *et al.*, "Progressive point set surfaces," *ACM Trans. Graph.,* vol. 22, no. 4, pp. 997-1011, 2003.

[24] F. Bernardini, J. Mittleman, H. Rushmeier *et al.*, "The Ball-Pivoting Algorithm for Surface Reconstruction," *IEEE Transactions on Visualization and Computer Graphics,* vol. 5, no. 4, pp. 349-359, 1999.

[25] J. Huang, and C. H. Menq, "Combinatorial Manifold Mesh Reconstruction and Optimization from Unorganized Points with Arbitrary Topology," *Computer-Aided Design,* vol. 34, no. 2, pp. 149-165, 2002.

[26] H.-W. Lin, C.-L. Tai, and G.-J. Wang, “A Mesh Reconstruction Algorithm Driven by An Intrinsic Property of A Point Cloud,” *Computer-Aided Design,* vol. 36, no. 1, pp. 1-9, 2004.

[27] X. Li, C.-Y. Han, and W. G. Wee, “On Surface Reconstruction: A Priority Driven Approach,” *Computer-Aided Design,* vol. 41, no. 9, pp. 626-640, 2009.

[28] D. Cohen-Steiner, and F. Da, “A Greedy Delaunay-Based Surface Reconstruction Algorithm,” *The Visual Computer,* vol. 20, no. 1, pp. 4-16, 2004.

[29] C.-C. Kuo, and H.-T. Yau, “A New Combinatorial Approach to Surface Reconstruction with Sharp Features,” *IEEE Transactions on Visualization and Computer Graphics,* vol. 12, no. 1, pp. 73-82, 2006.

[30] J.-D. Boissonnat, “Geometric Structures for Three-Dimensional Shape Representation,” *ACM Transactions on Graphics,* vol. 3, no. 4, pp. 266-286, 1984.

[31] H. Edelsbrunner, and E. P. Mücke, “Three-Dimensional Alpha Shapes,” *ACM Transactions on Graphics,* vol. 13, no. 1, pp. 43-72, 1994.

[32] X. Xu, and K. Harada, “Automatic Surface Reconstruction with Alpha-shape Method,” *The Visual Computer,* vol. 19, no. 7, pp. 431-443, 2003.

[33] R. C. Veltkamp, “The γ-Neighborhood Graph,” *Computational Geometry,* vol. 1, no. 4, pp. 227-246, 1992.

[34] U. Adamy, J. Giesen, and M. John, “Surface Reconstruction Using Umbrella Filters,” *Computational Geometry,* vol. 21, no. 1-2, pp. 63-86, 2002.

[35] A. Rosner. "Curve Reconstruction," **http://valis.cs.uiuc.edu/~sariel/research/CG/**.

[36] N. Amenta, M. Bern, and M. Kamvysselis, “A New Voronoi-Based Surface Reconstruction Algorithm,” in Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, 1998, pp. 415-421.

[37] N. Amenta, and M. Bern, “Surface Reconstruction by Voronoi Filtering,” *Discrete and Computational Geometry,* vol. 22, no. 4, pp. 481-504, 1999.

[38] N. Amenta, S. Choi, and R. K. Kolluri, “The Power Crust,” in Proceedings of the 6th ACM Symposium on Solid Modeling and Applications, Ann Arbor, Michigan, United States, 2001, pp. 249-266.

[39] N. Amenta, S. Choi, and R. K. Kolluri, “The power crust, unions of balls, and the medial axis transform,” *Computational Geometry,* vol. 19, no. 2-3, pp. 127-153, 2001.

[40] N. Amenta, S. Choi, T. K. Dey *et al.*, “A simple algorithm for homeomorphic surface reconstruction,” in Proceedings of the sixteenth annual symposium on

Computational geometry, Clear Water Bay, Kowloon, Hong Kong, 2000, pp. 213-222.

[41] T. K. Dey, and J. Giesen, "Detecting undersampling in surface reconstruction," in Proceedings of the seventeenth annual symposium on Computational geometry, Medford, Massachusetts, United States, 2001, pp. 257-263.

[42] T. K. Dey, and S. Goswami, "Provable surface reconstruction from noisy samples," in Proceedings of the twentieth annual symposium on Computational geometry, Brooklyn, New York, USA, 2004, pp. 330-339.

[43] CGAL. "Computational Geometry Algorithm Library," **http://www.cgal.org**.

[44] K. R. Gabriel, and R. R. Sokal, "A New Statistical Approach to Geographic Variation Analysis," *Systematic Biology,* vol. 18, no. 3, pp. 259-278, September 1, 1969, 1969.

[45] R. C. Veltkamp, "3D Computational Morphology," *Computer Graphics Forum,* vol. 12, no. 3, pp. 115-127, 1993.

[46] M. Attene, and M. Spagnuolo, "Automatic Surface Reconstruction from Point Sets in Space," *Computer Graphics Forum,* vol. 19, no. 3, pp. 457-465, 2000.

[47] VCG. "VCG Library," **http://vcg.sourceforge.net**.

[48] H.-T. Yau, C.-C. Kuo, and C.-H. Yeh, "Extension of Surface Reconstruction Algorithm to The Global Stitching and Repairing of STL models," *Computer-Aided Design,* vol. 35, no. 5, pp. 477-486, 2003.

[49] A. Adamson, and M. Alexa, "Ray Tracing Point Set Surfaces," in Proceedings of Shape Modeling International, 2003, pp. 272-279.

[50] N. Amenta, and Y. J. Kil, "Defining Point-set Surfaces," *ACM Trans. Graph.,* vol. 23, no. 3, pp. 264-270, 2004.

[51] T. K. Dey, and J. Sun, "An adaptive MLS Surface for Reconstruction with Guarantees," in Proceedings of the third Eurographics symposium on Geometry processing, Vienna, Austria, 2005, pp. 43.

[52] R. Hoffman, and A. K. Jain, "Segmentation and Classification of Range Images," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 9, no. 5, pp. 608-620, 1987.

[53] M. J. Milroy, C. Bradley, and G. W. Vickers, "Segmentation of A Wrap-around Model Using An Active Contour," *Computer-Aided Design,* vol. 29, no. 4, pp. 299-320, 1997.

[54] M. Yang, and E. Lee, “Segmentation of Measured Point Data Using A Parametric Quadric Surface Approximation,” *Computer-Aided Design,* vol. 31, no. 7, pp. 449-457, 1999.

[55] H. Jianbing, and M. Chia-Hsiang, “Automatic Data Segmentation for Geometric Feature Extraction from Unorganized 3-D Coordinate Points,” *Robotics and Automation, IEEE Transactions on,* vol. 17, no. 3, pp. 268-279, 2001.

[56] H. Woo, E. Kang, S. Wang *et al.*, “A New Segmentation Method for Point Cloud Data,” *International Journal of Machine Tools and Manufacture,* vol. 42, no. 2, pp. 167-178, 2002.

[57] G. Schaufler, and H. W. Jensen, “Ray Tracing Point Sampled Geometry,” in Proceedings of the Eurographics Workshop on Rendering Techniques 2000, 2000, pp. 319-328.

[58] T. K. Dey, G. Li, and J. Sun, “Normal Estimation for Point Clouds: A Comparison Study for a Voronoi Based Method,” in Eurographics Symposium on Point-Based Graphics, 2005, pp. 39-46.

[59] N. J. Mitra, A. Nguyen, and L. Guibas, “Estimating Surface Normals in Noisy Point Cloud Data,” *International Journal of Computational Geometry and Applications,* vol. 14, no. 4-5, pp. 261-276, 2004.

[60] M. Pauly, R. Keiser, L. P. Kobbelt *et al.*, “Shape Modeling with Point-sampled Geometry,” *ACM Trans. Graph.,* vol. 22, no. 3, pp. 641-650, 2003.

[61] D. OuYang, and H. Y. Feng, “On the normal vector estimation for point cloud data from smooth surfaces,” *Computer-Aided Design,* vol. 37, no. 10, pp. 1071-1079, 2005.

[62] J. O'Rourke, *Computational Geometry in C*: Cambridge University Press, 1994.

[63] R. C. Veltkamp, *Closed Object Boundaries From Scattered Points*, Secaucs, NJ, USA: Springer-Verlag New York, 1994.

[64] J. Ma, H. Y. Feng, and L. Wang, “Delaunay-Based Triangular Surface Reconstruction from Points via Umbrella Facet Matching,” in Proceedings of the 6th IEEE Conference on Automation Science and Engineering, 2010, pp. 580-585.

[65] H. S. M. Coxeter, *Introduction to Geometry*, 2nd ed.: Wiley, 1989.

[66] D. J. Struik, *Lectures on Classical Differential Geometry*, 2nd ed., MA: Addison-Wesley, 1961.

[67] E. Zurich. "Pointshop3D," **http://graphics.ethz.ch/pointshop3d/**.

[68] J. Han, M. Pratt, and W. C. Regli, “Manufacturing Feature Recognition from Solid Models: A Status Report,” *IEEE Transactions on Robotics and Automation,* vol. 16, no. 6, pp. 782-796, 2000.

[69] C. Weber, S. Hahmann, and H. Hagen, “Sharp Feature Detection in Point Clouds,” in Shape Modeling International Conference (SMI), 2010, pp. 175-186.

[70] M. Attene, B. Falcidieno, J. Rossignac *et al.*, “Sharpen&Bend: Recovering Curved Sharp Edges in Triangle Meshes Produced by Feature-Insensitive Sampling,” *IEEE Transactions on Visualization and Computer Graphics* vol. 11, no. 2, pp. 181-192, 2005.

[71] F. Bernardini, C. L. Bajaj, J. Chen *et al.*, “A triangulation-based object reconstruction method,” in Proceedings of the thirteenth annual symposium on Computational geometry, Nice, France, 1997, pp. 481-484.

[72] W. E. Lorensen, and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *SIGGRAPH Comput. Graph.,* vol. 21, no. 4, pp. 163-169, 1987.

[73] L. P. Kobbelt, M. Botsch, U. Schwanecke *et al.*, “Feature sensitive surface extraction from volume data,” in Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 2001, pp. 57-66.

[74] Y. Ohtake, A. Belyaev, and A. Pasko, “Dynamic Meshes for Accurate Polygonization of Implicit Surfaces with Sharp Features,” in Shape Modeling and Applications, SMI 2001 International Conference on., 2001, pp. 74-81.

[75] G. Casciola, D. Lazzaro, L. B. Montefusco *et al.*, “Shape preserving surface reconstruction using locally anisotropic radial basis function interpolants,” *Computers & Mathematics with Applications,* vol. 51, no. 8, pp. 1185-1198, 2006.

[76] X. Wang, and R. Macleod, “Feature Extraction from Point Clouds,” *Proceedings, 10th International Meshing Roundtable*, 2001.

[77] K. Demarsin, D. Vanderstraeten, T. Volodine *et al.*, “Detection of closed sharp edges in point clouds using normal estimation and graph theory,” *Computer-Aided Design,* vol. 39, no. 4, pp. 276-283, 2007.

[78] H. Song, H. Y. Feng, and D. OuYang, “Automatic Detection of Tangential Discontinuities in Point Cloud Data,” *Journal of Computing and Information Science in Engineering,* vol. 8, no. 2, pp. 021001-10, 2008.

[79] S. Fleishman, D. Cohen-Or, and C. T. Silva, “Robust Moving Least-squares Fitting with Sharp Features,” *ACM Transactions on Graphics,* vol. 24, no. 3, pp. 544-552, 2005.

[80] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression," *Computer Graphics Forum,* vol. 28, no. 2, pp. 493-501, 2009.

[81] R. Mencl, and H. Muller, "Graph-Based Surface Reconstruction Using Structures in Scattered Point Sets," in Proceedings of Computer Graphics International, 1998, pp. 298-311.

[82] A. Hubeli, and M. Gross, "Multiresolution Feature Extraction from Unstructured Meshes," in Proceedings of Visualization 2001, pp. 287-294.

[83] K. Watanabe, and A. G. Belyaev, "Detection of Salient Curvature Features on Polygonal Surfaces," *Computer Graphics Forum,* vol. 20, no. 3, pp. 385-392, 2001.

[84] T. K. Dey. **http://www.cse.ohio-state.edu/%7Etamaldey/**.

CURRICULUM VITAE

Name:	Ji Ma
Education:	B.E., 1989-1993 Aeronautics Engineering Beijing University of Aeronautics & Astronautics Beijing, P. R. China
	M.E., 1995-1998 Mechanical & Materials Engineering Wuhan University of Technology Wuhan, Hubei, P. R. China
	Ph.D., 2006-2011 Mechanical & Materials Engineering The University of Western Ontario London, Ontario, Canada
Experience:	Aeronautics Engineer No. 605 Institute of Aviation Industries of China 1993 – 1995
	Senior Lecturer and Mechanical Engineer Wuhan University of Technology 1998 - 2005
	Senior Software Engineer (Part-time) Wuhan Kangbo Software Corporation 2001 - 2005

Refereed Journal Articles:

1. Ma, J., Feng, H. Y., and Wang, L., "Watertight Manifold Triangle Mesh Reconstruction from Points via Umbrella Facet Matching," *Computer-Aided Design* (submitted, June 2011).

2. Endrias, D. H., Feng, H. Y., Ma, J., Wang, L., and Taher, M. A., "A Combinatorial Optimization Approach for Evaluating Minimum-Zone Spatial Straightness Errors," *Measurement* (submitted, February 2011).

3. Wang, L., Cai, N., Feng, H. Y., and Ma, J., 2010, "ASP: An Adaptive Setup Planning Approach for Dynamic Machine Assignments," *IEEE Transactions on Automation Science and Engineering*, Vol. 7, No. 1, pp. 2-14.

Conference Papers:

1. Ma, J., Feng, H. Y., and Wang, L., "Feature Sensitive Mesh Reconstruction by Normal Vector Cone Filtering," *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Paper #DETC2011-47410, Washington, DC, August 29-31, 2011 (accepted).

2. Ma, J., Feng, H. Y., and Wang, L., 2010, "Delaunay-Based Triangular Surface Reconstruction from Points via Umbrella Facet Matching," *Proceedings of the 6th IEEE Conference on Automation Science and Engineering*, pp. 580-585, Toronto, Canada, August 21-24, 2010.

3. Wang, L., Ma, J., and Feng, H. Y., 2009, "Web-DPP: A Distributed Process Planning System for Adaptive Machining Operations," *Proceedings of the 19th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2009)*, pp. 186-193, Middlesbrough, UK, July 6-8, 2009.

4. Wang, L., Ma, J., and Feng, H. Y., 2008, "An Adaptive and Optimal Setup Planning System," *Proceedings of the 4th IEEE Conference on Automation Science and Engineering*, pp. 67-72, Washington DC, August 23-26, 2008.

Book Chapter:

1. Wang, L., Feng, H. Y., Cai, N., and Ma, J., 2009, "Adaptive Setup Planning for Job Shop Operations under Uncertainty," *Collaborative Design and Planning for Digital Manufacturing* (L. Wang and A. Y. C. Nee, eds., ISBN 978-1-84882-286-3), pp. 187-216, Springer-Verlag, London, UK.