การเติมพื้นผิววัตถุให้สมบูรณ์ด้วยวิธีการแปลงลาปลาเชี่ยน

นายพงศกรณ์ วิจิตเวชไพศาล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

SURFACE COMPLETION USING LAPLACIAN TRANSFORM

Mr.Pongsagon Vichitvejpaisal

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2011

Thesis Title	SURFACE COMPLETION USING LAPLACIAN TRANSFORM
By	Mr.Pongsagon Vichitvejpaisal
Field of Study	Computer Engineering
Thesis Advisor	Assistant Professor Pizzanu Kanongchaiyos, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

..... Dean of the Faculty of Engineering

(Associate Professor Boonsom Lerdhirunwong, Dr.Ing.)

THESIS COMMITTEE

..... Chairman

(Associate Professor Somchai Prasitjutrakul, Ph.D.)

..... Thesis Advisor

(Assistant Professor Pizzanu Kanongchaiyos, Ph.D.)

..... External Examiner

(Associate Professor Pavadee Sompagdee)

..... External Examiner

(Associate Professor Natasha Dejdumrong, D.Tech.Sci.)

..... External Examiner

(Chakrit Watcharopas, Ph.D.)

พงศกรณ์ วิจิตเวชไพศาล : การเติมพื้นผิววัตถุให้สมบูรณ์ด้วยวิธีการแปลงลาปลาเซี่ยน. (SURFACE COMPLETION USING LAPLACIAN TRANSFORM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. พิษณุ คนองชัยยศ, 88 หน้า.

หลายครั้งที่การได้มาซึ่งพื้นผิววัตถุมีความไม่สมบูรณ์ เนื่องจากข้อจำกัดทางด้านเทคนิคต่างๆ วิทยานิพนธ์ฉบับนี้ได้นำเสนออัลกอริทึมในการเติมพื้นผิววัตถุให้สมบูรณ์โดยใช้บริบทของพื้นผิวที่มีอยู่แล้วจากตัวอย่าง งานวิจัยทางด้านการเติมพื้นผิววัตถุฉบับก่อนๆ ยังไม่สามารถรองรับวัตถุที่มีลักษณะพื้นผิวที่มีลวดลายซ้ำกันแบบสม่ำเสมอหรือไกล้เคียงได้ดี เป้าหมายของงานวิจัยชิ้นนี้คือ การสังเคราะห์พื้นผิวให้กับวัตถุที่มีรู โดยพื้นผิวที่สังเคราะห์จะต้องมีลักษณะบริบทของพื้นผิวเหมือนกับลักษณะของพื้นโดยรอบรู งานวิจัยชิ้นนี้สามารถรองรับวัตถุที่พื้นผิวมีลวดลายแบบสม่ำเสมอ แบบไม่สม่ำเสมอ หรือแบบสุ่มได้ งานวิจัยได้ใช้หลักการแยกรายละเอียดวัตถุเพื่อแยกวัตถุออกเป็นสองส่วน คือ ส่วนความละเอียดต่ำกับส่วนความละเอียดสูง ส่วนความละเอียดต่ำจะถูกเติมเต็มแบบเรียบ ส่วนความละเอียดสูงจะถูกทำการแปลงลาปลาเซี่ยนและถูกเติมเต็มด้วยวิธีการสังเคราะห์เชิงตัวอย่าง ระบบพิกัดลาปลาเซี่ยนเป็นการเก็บข้อมูลเส้นปกติและความโค้งของแต่ละจุดบนพื้นผิววัตถุ และจะถูกใช้เป็นลายเซ็นต์ของพื้นผิวในการตรวจสอบการเหมือนกันของส่วนของพื้นผิว หลังจากส่วนของวัตถุทั้งสองส่วนได้ถูกเติมเต็มแล้ว จะถูกนำมารวมกันในระบบพิกัดลาปลาเซี่ยน และทำการแปลงกลับเพื่อสร้างพื้นผิวที่สมบูรณ์ อัลกอริทึมของงานวิจัยฉบับนี้ได้ทำการทดลองกับพื้นผิวแบบเรียบ พื้นผิวแบบโค้ง ที่มีลักษณะลวดลายพื้นผิวทุกรูปแบบ ผลการทดลองแสดงให้เห็นว่า รูสามารถถูกเติมให้เต็มด้วยพื้นผิวสังเคราะห์ที่มีลักษณะลวดลายเหมือนกับพื้นผิวต้นฉบับโดยรอบรู โดยรอยต่อระหว่างพื้นผิวเดิมกับพื้นผิวสังเคราะห์ไม่มีให้ปรากฏ

ภาควิชา.... วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....
สาขาวิชา....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่ออ.ที่ปรึกษาวิทยานิพนธ์หลัก.....
ปีการศึกษา .....2554.....

4971871921: MAJOR COMPUTER ENGINEERING
KEYWORDS: SURFACE COMPLETION / LAPLACIAN TRANSFORM / EXAMPLE-BASED SYNTHESIS / MULTIRESOLUTION DECOMPOSTION

PONGSAGON VICHITVEJPAISAL: SURFACE COMPLETION USING LAPLACIAN TRANSFORM. ADVISOR: ASST. PROF. PIZZANU KANONGCHAIYOS, Ph.D., 88 pp.

Model acquisition process usually produce incomplete surfaces due to the technical constrains. This thesis presents the algorithm to perform surface completion using the available surface's context. Previous works on surface completions do not handle surfaces with near-regular pattern or irregular patterns well. The main goal of this research is to synthesize surface for hole that will have similar surface's context or geometric details as the hole's surrounding. The surfaces can have near-regular patterns, irregular patterns or stochastic patterns. This research uses multi-resolution approach to decompose the model into low-frequency part and high-frequency part. The low-frequency part is filled smoothly. The high-frequency part are transformed it into the Laplacian coordinate and filled using example-based synthesize approach. Laplacian coordinate, which encodes normal and curvature of each point on the surface, is used as the surface signature when perform geometric detail similarity search. After all the two resolution parts is filled, they are combined in Laplacian coordinate and are transformed using inverse Laplacian transform to reconstruct the complete surface. The algorithm is tested with planar surfaces and curve surfaces with all kind of relief patterns. The results indicate that the holes can be completed with the geometric detail similar to the surrounding surface. The seam between the hole and the surrounding surface is hardly visible.

Department : …Computer Engineering… Student's Signature.....
Field of Study : ..Computer Engineering...Advisor's Signature.....
Academic Year : …2011……..

ACKNOWLEDGEMENTS

It is a great honor to acknowledge my thesis advisor, Assistant Professor Pizzanu Kanongchaiyos, Ph.D., for his intellectual advices and invariable assistances throughout this research. I would also like to express my grateful thanks to my thesis committee, Associate Professor Somchai Prasitjutrakul, Ph.D., Associate Professor Pavadee Sompagdee, Associate Professor Natasha Dejdumrong, D.Tech.Sci., Chakrit Watcharopas, Ph.D. for their beneficial guidance and suggestions. I would like to thank H.M. the King's 72nd Birthday Scholarship and The Royal Golden Jubilee Ph.D Program Scholarship. I want to extend my thanks to all research members especially my associates in computer graphic lab (CG Lab) for their generous helps, encouragements and relationships which make my life through the course filled with amusements and happiness.

Finally, I deeply wish to thank my parents for their love, understanding and invaluable supports throughout my graduate study.

CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Page

Figure

Page

Figure

CHAPTER I

INTRODUCTION

1.1 Background and Statement of Problems

In recent years, we have seen widely spread use of 3D model acquisition systems. The 3D scanning techniques are becoming to go beyond professional projects to mass public usages. However, the acquired surface usually incomplete due to many reasons such as noise, limited viewpoints, self-occlusion, and technological constrains. The incompleteness of the surface or holes is needed to be filled before the further use of the acquired model. Furthermore, some model editing operation such as cutting and pasting introduce holes to the model and thus these holes need to be filled automatically. In order for 3D scanning systems to be widely available to the consumer, robust and easy-to-use surface completion methods need to be developed for the users.

Surfaces can be categorized into smooth surfaces and non-smooth surfaces. Smooth surfaces are surfaces with low geometric variations. There are no geometric details or relief information on the surfaces. On the other hand, non-smooth surfaces exhibits relief information or geometric patterns on the surfaces. In this work, the relief patterns of non-smooth surfaces are classified as near-regular patterns, irregular patterns and stochastic patterns (Figure 1.1).

Near-regular patterns are patterns that obviously have repetition of the geometric elements. The repetition may have some distortions or offsets. Irregular patterns are patterns that have some geometric structures. The structures look similar to each other but they are not the replications of each other. Stochastic patterns are patterns that do not exhibit geometric structures. However, stochastic patterns have some statistical distribution of geometric properties.

The definition of a hole is subjective to the user's view. Usually, the user selects a part of the surface to perform surface completion. In this work, a hole is defined as a closed cycle of boundary edges [1]. In some cases, a hole may have isles on it. The isles are the fragment of the surface.

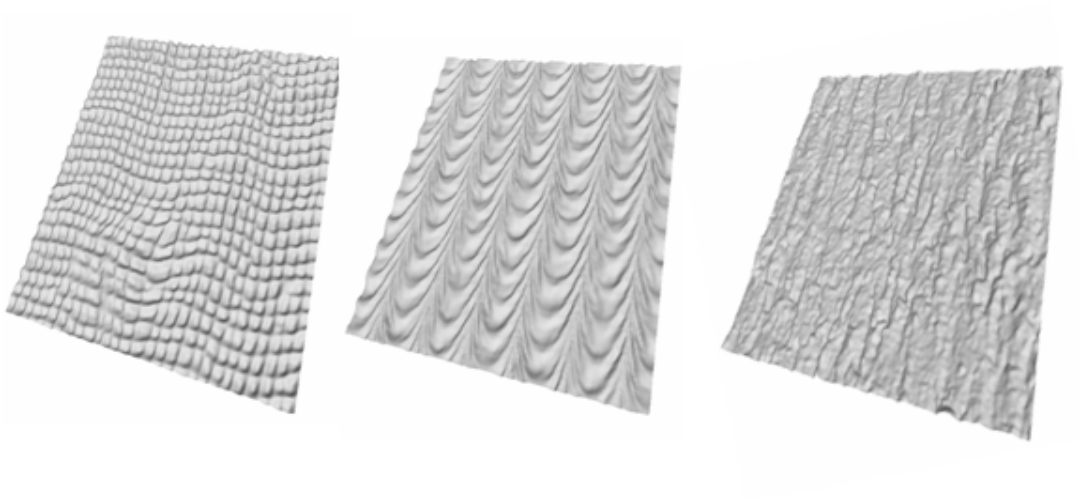

Figure 1.1: An example of near-regular, irregular and stochastic relief patterns.

Researches on surface completion can be divided into two groups.

The first group [2-4] fills hole on the model smoothly (Figure 1.2 (b)). These methods usually rely on boundary conditions or algorithms based on diffusion process to fill the hole. However, for surfaces that have geometric detail, it can be easily noticed that the filled surface is not consistence with the surrounding surface. Thus, the users have to painstakingly edit the filled surface to make it match the surrounding surface.

As a matter of fact, the second group [5-12] try to analyze the geometric or surface's relief information available in the input surface and transfer this features to the hole's surface (Figure 1.2 (c)). These context-based methods attempt to produce the hole's surface that has similar geometric detail as the input surface. Unfortunately, analyzing relief pattern and transferring it to the hole's surface is not trivial. Usually, most of the previous works [8-12] can handle patterns that are stochastic. However, relief patterns that contain near-regular or irregular structures are still a challenging problem.

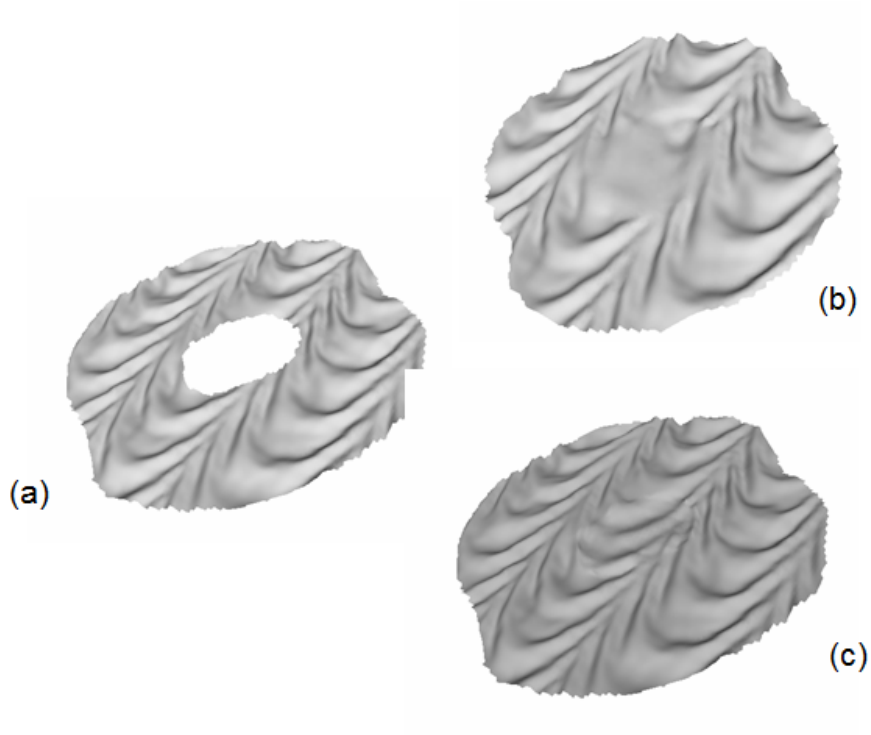

Figure 1.2: Smooth surface completion (b) and Context-based surface completion (c) of the input surface (a).

This work presents a method for completing the mesh surfaces that contain near-regular, irregular or stochastic relief patterns. The algorithm is designed to analyze and extract these features from the input mesh and fill it to the hole. In addition, the algorithm attempts to construct the hole's surface with overall structure that is similar to the input surface.

This research is built on the two ideas, multi-resolution decomposition of meshes and example-based synthesis.

The multi-resolution decomposition [13] views the surface as compose of low-frequency part which represents the overall surface structure and high-frequency part which represents the geometric surface detail. This view, in many ways, reflects the human observation on object shapes. The technique is used extensively in model editing process [14]. Artists can edit one part without disturbing the other part.

This research decomposes the surface into two parts, the coarse mesh and the relief mesh. First, the hole of the coarse mesh is smoothly filled. Then, the relief pattern is transferred to this hole. This is to ensure that both the structure of the filled hole and the relief pattern on the hole are consistent with the input surfaces.

The example-based synthesis framework [15-16] is originated from the 2D texture synthesis field. It is used to synthesis a new texture that has visual similarity with the input texture but is larger in size. Similarity is defined by the color difference between neighborhoods of the two pixels. The value of each output pixel is determined by comparing its spatial neighborhood with all pixels' neighborhoods of the input texture. The input pixel with the most similar neighborhood will be assigned to the corresponding output pixel. The results [17] of example-based synthesis are pleasing. They contain visual correspondence as presented in the input textures.

This research adapts the idea of the example-based framework to transfer the relief pattern to the smoothly filled hole. However, for the mesh domain, some important aspects need to be solved. First, mesh topologies do not align regularly in uniform grid style as in images. Meshes that are similar in shape may differ in topology considerably making the comparison between the two vertices' neighborhoods impossible. Second, surface similarity metric or surface signature is still an ongoing research. Usually, works on surface signature [8,10] measure only the overall likeness of the surfaces' shapes. Thus, they are not accurate enough to use in the example-based framework.

This research proposed the use of Laplacian coordinate as the representation for the coarse mesh and the relief mesh. Laplacian coordinate represents the curvature and normal of each point on the surface instead of the position. Thus, it looks promising to use as a surface signature for similarity test. The merging of the smoothly filled hole and the relief pattern is also done in Laplacian space.

1.2 Objectives of Study

The objective of this study is to propose an algorithm to fill the hole on mesh surfaces. The filled surface will contains the geometric surface detail or relief detail as presented in the existing surface. In addition, this study also proposes the

Laplacian based surface signature to compare the similarity between two surface patches.

1.3 Scopes of Study

1 The proposed algorithm can accept arbitrary 2-mainfold oriented surface mesh models.

2 The input models must compose of regular polygons, otherwise the input models have to be remeshed before use.

3 The input hole is defined as a single closed loop of boundary vertices. The algorithm does not accept hole with isles or disconnected hole boundary.

4 The input models should contain near-regular, irregular or stochastic relief details in order to guide the algorithm.

1.4 Research Procedures

1 Research and study previous works on mesh processing and surface completion. Analysis the advantages and disadvantages of each works.

2 Design the algorithm using divide and conquer technique.

3 The process of the algorithm is divided into many steps

   a. Finding the offset regions

   b. Performing mesh smoothing

   c. Completing coarse mesh

   d. Computing Laplacian coordinate

   e. Computing mesh parameterization

   f. Transferring relief pattern

   g. Computing inverse Laplacian transform.

4 Implement the algorithm.

5 Set up the experimentation with various test cases.

6 Test, improve and correct the algorithm.

7 Analysis and evaluate the proposed algorithm.

8 Do the conclusion, suggestions and plan the future work

1.5 Expected Benefits

1 The proposed method performs surface completion for general mesh models. The filled surfaces contain relief detail similar to the existing surface. Thus, the proposed method can save a considerable amount of time in the model acquisition process.

1.6 Thesis Structure

This thesis has five chapters namely - introduction, theoretical background and related works, the proposed method, experimental results, summary and future works.

Chapter 2 gives a brief description of the mesh definition, curvature definition, mesh representation and Laplacian representation framework. Related works on surface completion and example-based synthesis are also discussed. Chapter 3 presents the proposed algorithm. The algorithm can be divided into four parts – mesh decomposition, coarse mesh completion, relief mesh completion and combining of coarse mesh and relief mesh. In chapter 4, the algorithm is tested with various test cases and parameter settings. The results and the limitations of the algorithm are demonstrated. Thesis summary and future works are presented in chapter 5.

CHAPTER II

THEORETICAL BACKGROUND AND RELATED WORKS

2.1 Theoretical Background

2.1.1 Mesh Definition

In this work, 3D models are represented with polygon meshes. Although there are other model representation such as spline surfaces, subdivision surfaces and implicit surfaces, mesh representation can approximate wide range of surfaces and does not have the continuity constrain as the other surfaces. Mesh representation is simple, robust and flexible surface representation. There are many algorithms [18-20] that convert other surface representations to mesh.

This research is also restricted mesh representation to compose of only triangles. Since triangle is the only polygon that can guarantee its vertices to be on the same plane. Triangle requires less time to manipulate than other type of polygons and can be directly processed by graphics hardware. Other polygon types can be converted to triangle with the use of triangulation algorithms [21-22].

Some properties of the triangle mesh are defined as followed:

For every vertex **v**, the **valence** of a vertex is defined as the number of neighborhood vertex that has direct edge to it.

Triangle is **isotropic** if its shape is close to equilateral triangle. For example, the long thin triangle has low isotropic value.

Mesh has **uniform distribution** if the mesh elements are evenly spread across the entire model.

Mesh topology is the interconnection pattern of edges and vertices.

The **interior edge** is an edge that is share by two triangles.

The **boundary edge** is an edge that connects to only one triangle.

The **interior vertex** is a vertex that is connected only by interior edge.

The **boundary vertex** is a vertex that has one or more boundary edge.

A vertex in a triangle mesh is **regular** if its valence is 6 for interior vertices or 4 for boundary vertices.

A surface is a 2-manifold if it is everywhere locally homeomorphic to a disk.

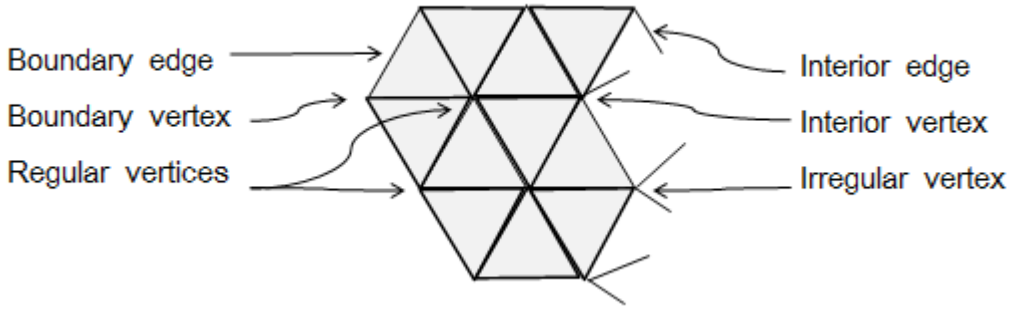

Figure 2.1: An example of triangle mesh with some topology definitions.

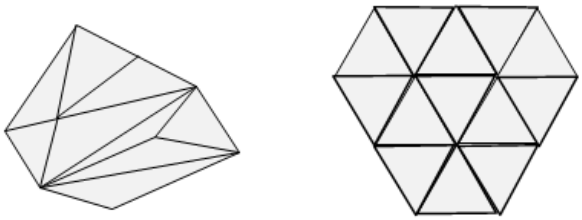

Figure 2.2: (Left) Mesh with low isotropic triangles and (Right) mesh with high isotropic triangles.

2.1.2 Curvature Definition

This section shows the curvature definition on surfaces. Curvature plays an important role on mesh processing and surface analysis. The applications are such as mesh smoothing, mesh parameterization, mesh editing and mesh deformation. Curvatures are the basis local properties of the surfaces. The definitions are defined for continuous surface. For discrete surfaces such as meshes, approximation formulas are required.

Let a continuous surface $S \subset R^3$ be given in parametric form as

$$x(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, (u,v) \in R^2$$

Normal vector is given as

$$n = \frac{(x_u \times x_v)}{\left\| x_u \times x_v \right\|}$$

In mesh representation, normal vector of a polygon is usually computed from the cross product between two polygon's edges.

The **first fundamental** form of x is

$$I = \begin{bmatrix} x_u^T x_u & x_u^T x_v \\ x_u^T x_v & x_v^T x_u \end{bmatrix}$$

The **second fundamental** form is

$$II = \begin{bmatrix} x_{uu}^T n & x_{uv}^T n \\ x_{uv}^T n & x_{vv}^T n \end{bmatrix}$$

Let $t = ax_u + bx_v$ be a unit vector in the tangent plane at P, represented as $\bar{t} = (a,b)^T$ (Figure 2.3)

Normal curvature in direction $\bar{t}$ can be expressed as

$$k_n(\bar{t}) = \frac{\bar{t}^T II \bar{t}}{\bar{t}^T I \bar{t}}$$

Principal curvatures are used to define the curvature at each point on the surface and do not have to depend on the chosen direction $\bar{t}$. Principal curvatures are minimal curvature and maximal curvature.

Minimal curvature is defined as

$$k_1 = k_{\min} = \min_{\bar{t}} k_n(\bar{t})$$

Maximal curvature is defined as

$$k_2 = k_{\max} = \max_{\bar{t}} k_n(\bar{t})$$

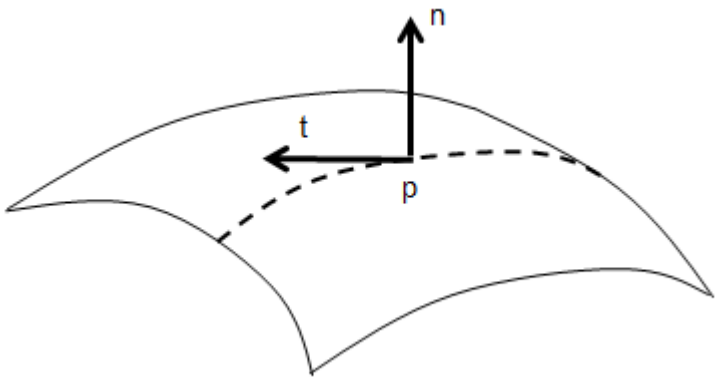

Figure 2.3: At point p of the surface x, the normal vector n and tangent vector t are perpendicular to each other. The dash line shows the intersection of surface x and the tangent plane defined by vector n and t. The normal curvature is defined for each point and for each direction t.

Beside principal curvatures, there are two other types of curvature, Gaussian and Mean curvature, which are defined using minimal and maximal curvature.

Gaussian curvature is defined as equation (1) and Mean curvature is defined as equation (2).

$$K = k_1 k_2 \quad (1)$$

$$H = \frac{k_1 + k_2}{2} \quad (2)$$

Mean curvature can be approximated for each point on mesh using Laplacian operator as presented in section 2.1.4.1.

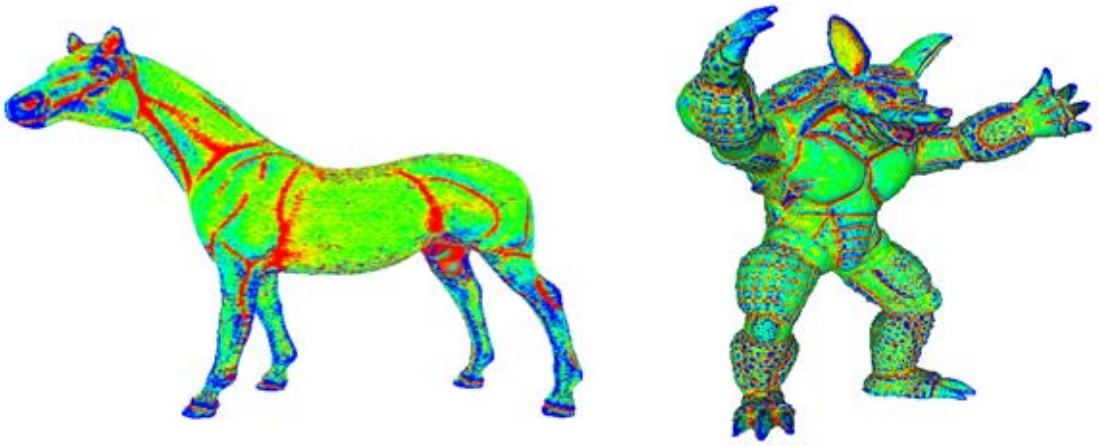

Figure 2.4: The visualization of the Mean curvature of the horse model and the armadillo model. The blue color indicates the positive mean curvature value or surface with convex curvature. The red color indicates the negative mean curvature value or surface with concave curvature. The green color indicates planar surface.

2.1.3 Mesh Representation

For efficiency in computation, the Half-edge data structure is used in geometry processing. Then, it is converted to Indexed face representation that is more suitable for the rendering pipeline.

2.1.3.1 Indexed Face

Indexed face represents meshes by the used of vertex list and face list. Vertex list stores the vertices position while the face list stores the mesh triangle (face) vertices by using indexes number on the vertex list. The drawback of the indexed face representation is that it does not store the neighborhood information of each vertex explicitly. Thus, the neighborhood query operation of a vertex is O(n), where n is the number of vertex on mesh.

2.1.3.2 Half-Edge Data Structure

Half-Edge representation stores each edge twice [23-24]. Each half-edge stores a link to the vertex it points to, the next half-edge, its opposite half-edge

and its adjacent face. Although Half-Edge use more memory than Indexed face, this structure can perform neighborhood query such as 1-ring in O(1). As a matter of fact, it is used extensively in mesh processing field.

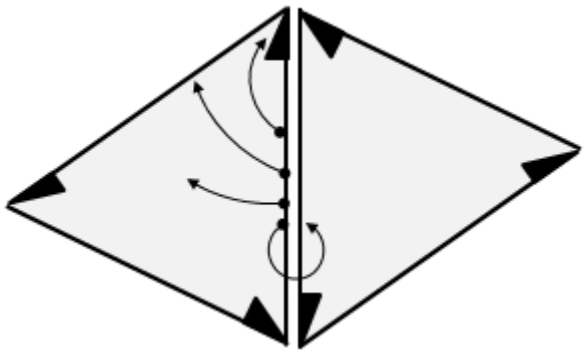

Figure 2.5: The references stores by a half-edge.

2.1.4 Laplacian Representation

2.1.4.1 Laplacian Coordinate and Laplacian Operator

Laplacian operator is defined as the divergence of the gradient and can be written as the sum of second partial derivatives [25].

$$\Delta f = div \nabla f = \sum_i \frac{\partial^2 f}{\partial x_i^2} \tag{3}$$

For a given function f defined on a manifold surface S the **Laplacian-Beltrami** is defined as

$$\Delta_S f = div_S \nabla_S f \tag{4}$$

Applied to the coordinate function x of the surface the Laplacian-Beltrami operator evaluates to the mean curvature normal

$$\Delta_S x = -2Hn \tag{5}$$

Thus, Laplacian representation encodes mean curvature multiply by normal vector of each point on the surface.

For discrete Laplacian-Beltrami operator, it can be approximated by Graph Laplacian or Geometric Mesh Laplacian.

Graph Laplacian [26] is defined as

$$\delta_i = (\delta_i^x, \delta_i^y, \delta_i^z) = \Delta_S x = \frac{1}{d_i} \sum_{j \in N(i)} (v_i - v_j) \tag{6}$$

where d_i is the number of immediate neighbor of vertex i. δ is called Laplacian coordinate. Laplacian coordinate is defined by the used of surface differential properties therefore it also called differential coordinates.

Geometric Mesh Laplacian [27] is defined as

$$\delta_i = \frac{1}{|\Omega_i|} \sum_{j \in N(i)} \frac{1}{2} (\cot \alpha_{ij} + \cot \beta_{ij})(v_i - v_j) \tag{7}$$

where Ω_i is the area of the Voronoi cell constructed around the faces of vertex i. In this research, Ω_i is equal to one-third of the area summation of all the faces around vertex i. $N(i)$ are the neighborhood vertex of i. α_{ij} and β_{ij} are the two angles the opposite to the edge ij.

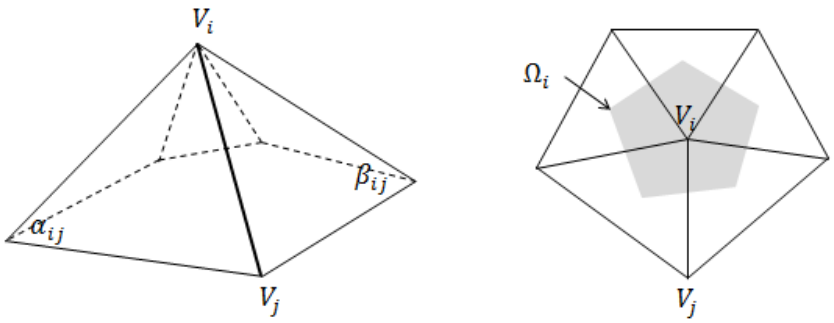

Figure 2.6: The angles and the Voronoi cell (shaded area) used for the Geometric Mesh Laplacian computation.

Graph Laplacian considers only the mesh's topological information while Geometric Mesh Laplacian also considers geometric information such as surface area

and angle. As a result, Geometric Mesh Laplacian produces more accurate Laplacian coordinate than Graph Laplacian method.

2.1.4.2 Laplacian Transform

Laplacian transform is the transformation from absolute coordinate which are Cartesian coordinate to Laplacian coordinate that represent local shape of the surface. The transformation can be represented in the matrix form [28].

Let $V = \{v_1,...,v_n\}$ be the set of vertices in absolute Cartesian coordinate.

$\Delta = \{\delta_i\}$ be the set of vertices in Laplacian coordinate.

A matrix L can be constructed such that $\Delta = LV$, that is $\delta_i = L(v_i)$. Matrix L can be viewed as the adjacency matrix of a graph of mesh. The dimension of the matrix L is nxn, where n is the number of mesh's vertices.

For Graph Laplacian, the matrix can be defined as

$$L_{ij} = \begin{cases} d_i & i = j \\ -1 & (i, j) \in E \\ 0 & otherwise \end{cases}$$

For Geometric Mesh Laplacian, the matrix can be defined as

$$w_{ij} = \frac{1}{2|\Omega_i|}(\cot\alpha_{ij} + \cot\beta_{ij})$$

$$L_{ij} = \begin{cases} -w_{ij} & (i, j) \in E \\ \sum_{j=1}^{n} w_{ij} & i = j \\ 0 & otherwise \end{cases}$$

E is the set of edges of mesh. The matrix is used to multiply each component (x, y, z) of Cartesian coordinate separately $Lx = \delta^x, Ly = \delta^y, Lz = \delta^z$.

2.1.4.3 Inverse Laplacian Transform

One property of the Laplacian coordinate is that it is translation invariant. Matrix L has rank = n-1 therefore its invert matrix cannot be computed.

To reconstruct the Cartesian coordinate, a position is assigned to a vertex in the matrix to get a full rank system and therefore has the unique solution [28]. The inverse Laplacian transform thus required solving a linear system of equation.

$$\delta_i = L(v_i) = v_i - \frac{1}{d_i} \sum_{j \in N_i} v_j \tag{8}$$

In practical application, such as model editing, the Cartesian coordinates are assigned to more than one vertex. These vertices are called anchor vertices. They act as hard constrains for the solver.

Let $C = \{1,2,...,m\}$ be the set of indices of those vertices that were assigned spatial coordinates $v_i = c_i, i \in C$.

This additional constrain makes the linear system over-determine. However, one can solve for a unique solution in the least-square sense. [29]

$$V' = \arg\min_{v'} \left(\left\| L(V') - \Delta \right\|^2 \right) \tag{9}$$

Additional rows of anchor vertices are added to the matrix L. For each of these rows, the column which its index is the anchor vertex's index has the value of 1 and 0 otherwise. This modified matrix is denoted as matrix $\tilde{L}$.

Also, the set of vertices in Laplacian coordinate, $\Delta = \{\delta_i\}$, has to be modified as set b to include the assigned spatial coordinates $b = \{\Delta, c_1, ..., c_m\}$.

Thus the linear system $\Delta = LV$ now becomes $\tilde{L}V = b$. To solve the least-square problem, the linear system $\tilde{L}V = b$ can be written in normal equation form as $(\tilde{L}^T \tilde{L})V = \tilde{L}^T b$. $\tilde{L}^T \tilde{L}$ is positive definite and sparse. This research uses a direct method called Cholesky factorization [30-32] to solver for this linear system.

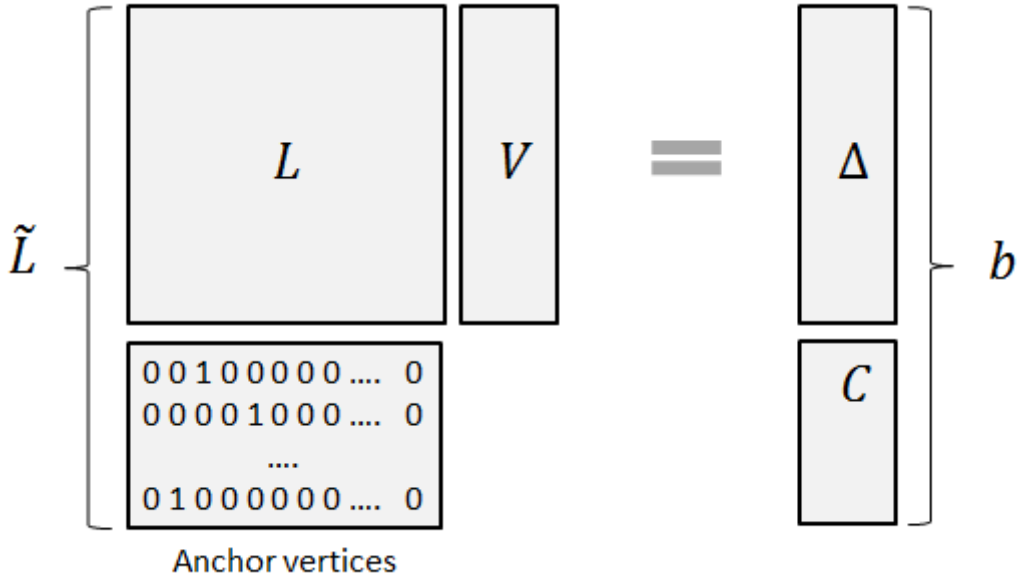

Figure 2.7: The components of the linear system $\tilde{L}V = b$ use in the Inverse Laplacian transform.

2.1.4.4 Features of Laplacian Coordinate

The most important features of the Laplacian coordinate is that it stores local information of Mean Curvature and Normal vector of each vertex. These are the local geometric properties of surface. Therefore, Laplacian representation can better reflect the local shape of the surface than the use of Cartesian representation.

The absolute coordinate reconstruction by linear least-squares method smoothly distributes the error across the domain. This feature is an advantage when perform mesh editing operation because distortion error is not dominant in some specific points on the surface.

Laplacian transform is translation invariant [29].

$$L(T(V)) = L(V) \tag{10}$$

Inverse Laplacian transform also has the rotational property of

$$L^{-1}(R(\Delta)) = R(L^{-1}(\Delta)) \tag{11}$$

T is the translation matrix and R is the rotation matrix.

2.2 Related Works

Surface completion can be broadly divided into two categories: one that does not consider contextual surface information and the other one that fill the hole with the available geometrical detail.

Since, the proposed algorithm of this thesis performs mesh smoothing and mesh parameterization. The related works of these methods are also presented in this part.

2.2.1 Smooth Surface Completion

Many authors proposed methods to smoothly fill the incomplete surfaces. Notably, Liepa [1] presents a method that can fill the hole smoothly and the filled surface preserved the overall structure of the model. The author use dynamic algorithm to fill the hole with subject to minimizing surface area.

Liepa's algorithm can be divided into three step namely – hole triangulation, mesh refinement [33] and fairing.

First, Hole triangulation finds a triangulation of the polygon that is defined by the hole boundary. The algorithm computes the triangulation that yields minimum triangulated surface area. In addition, the algorithm minimizes dihedral angles of all adjacent triangles in the triangulated surface.

The triangulated surface is refined so that the triangles in the triangulated surface are uniform and have the same size as the triangles in the offset region. Mesh refinement interactively splits long edges, collapse short edges, flips edges and relaxes vertices until the average edges length of the triangulated surface is equal to the edges length of the offset surface [34].

Finally, fairing is performs on the refined mesh [35]. Fairing is the method that minimizes surface curvature. The total curvature of a surface S can be expressed as the area integral of the sum of squared principal curvatures [36].

$$\int_S k_1^2 + k_2^2 dA \tag{12}$$

With the use of Variational calculus, the solution that minimizes principal curvatures can be written in the form of Bi-Laplacian equation.

$$\Delta^2 x = 0 \tag{13}$$

where x are the vertices positions.

Davis et al.[2] use the diffusion process to fill the hole. This method performs voxelization of the model and the cell is classified as inside, outside or on the surface boundary. Xie et al. [4] and Sharf et al.[3] solve the problem of surface reconstruction from point clouds by the use of active contour base method. The minimization technique is used to propagate the active contour to fit the given point clouds as much as possible. These methods can handle hole with small fragment and unconnected hole. However, the filled surface is smooth and lack of geometric details. For a large hole, the visual perception can be distracting. They also have to convert the model to volumetric representation which required computational time and memory consumption.

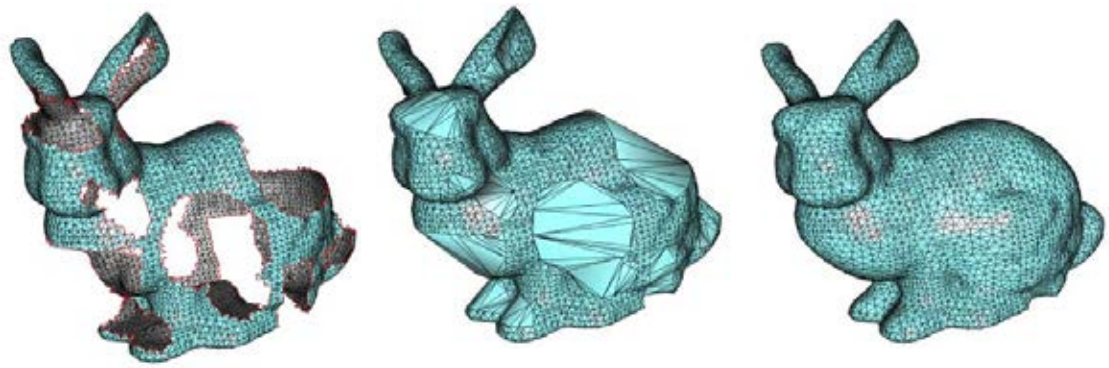

Figure 2.8: Smooth surface completion from [1]. The algorithm initially performs hole triangulation (middle) and follows by mesh refinement (right).

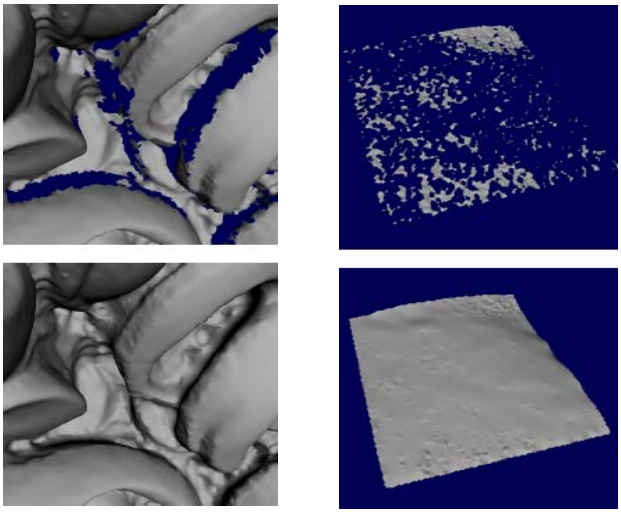

Figure 2.9: Volumetric based method [2] voxelizes the model and perform surface diffusion to fill the holes.

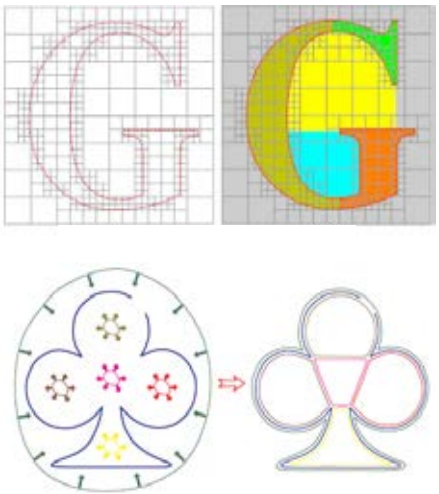

Figure 2.10: Active contour method [4] voxelizes the model and expand contours to fit the available surfaces. The surfaces of the expanded contours are used as the filled surfaces.

2.2.2 Surface Completion with Contextual Information

Many authors also consider contextual information when perform surface completion. Schnabel et al.[5] use primitives as the guidance for surface completion. They first fit the model with the predefined primitives and extended the primitives shape to fill the holes. Their works apply well for CAD model. Desbrun et al.[6] perform surface

completion using model database. Their methods search the database to find surface patches to fill the holes. The database has to contain the stock models that similar to the one it try to complete.

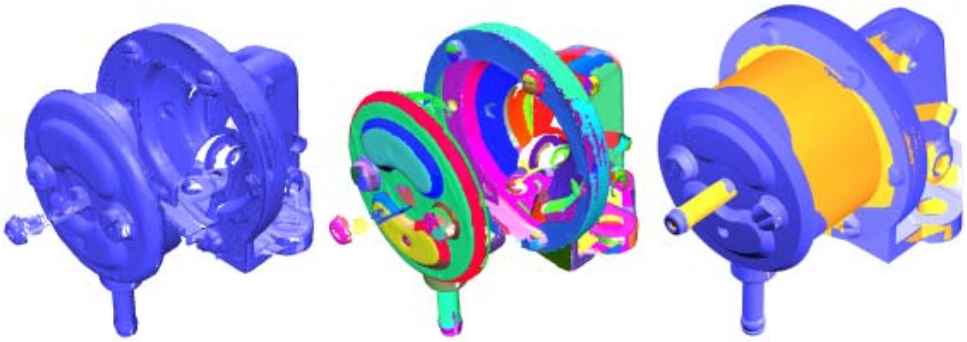

Figure 2.11: Primitives guidance method [5] finds the registration of the primitives to fit a given model.

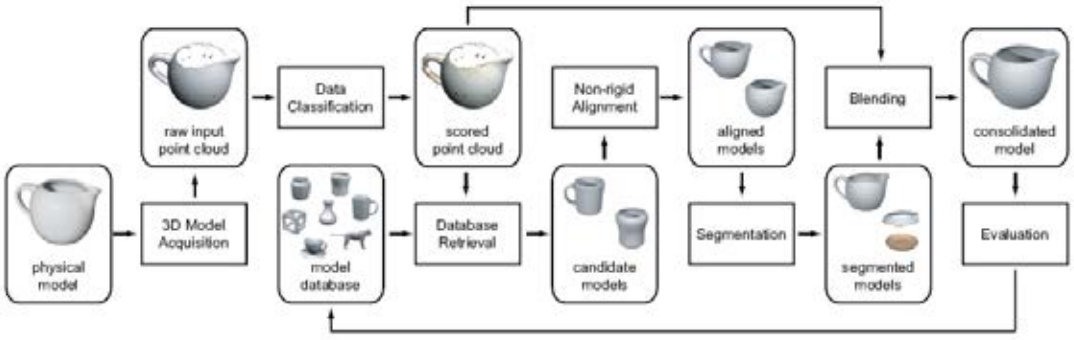

Figure 2.12: The pipeline of the Example-based 3D scan completion [6].

Shaft et al.[8] proposes an example-base method to do context filling of holes. They fill the missing surface iteratively from coarse to fine level using an octree. They use the signed distance vectors of grid corners as the surface signature for surface similarity test. The signature is a rough estimation of surface patterns which many not apply with near-regular or irregular pattern. Breckon et al.[7,11] also fills the hole from the available surface of the model. The method first fit the whole model with a primitive, such as a plane, sphere or cylinder. Then, the primitive is used as a base

surface to sample the vertex displacement vectors from the original surface. The vertex displacement vectors are used as surface signature to search for the similarity. This method may have difficulty with non-height field surfaces which cannot be represented by the vertex displacement vectors and the surface with steep surface detail can produce high distortion result from sampling. Bendels et al.[9] also use the vertex displacement vectors as in Breckon et al. However, instead of fitting the whole model to a primitive, they iteratively perform multi-level surface completion.

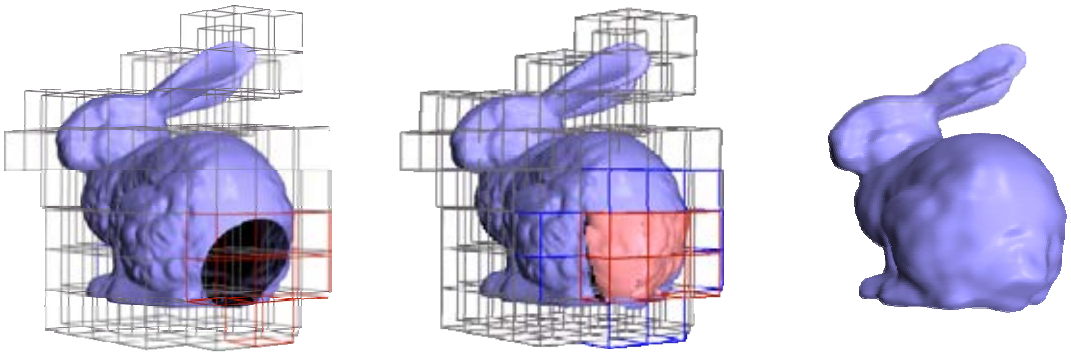

Figure 2.13: Context-based surface completion [8] fills the hole hierarchically.

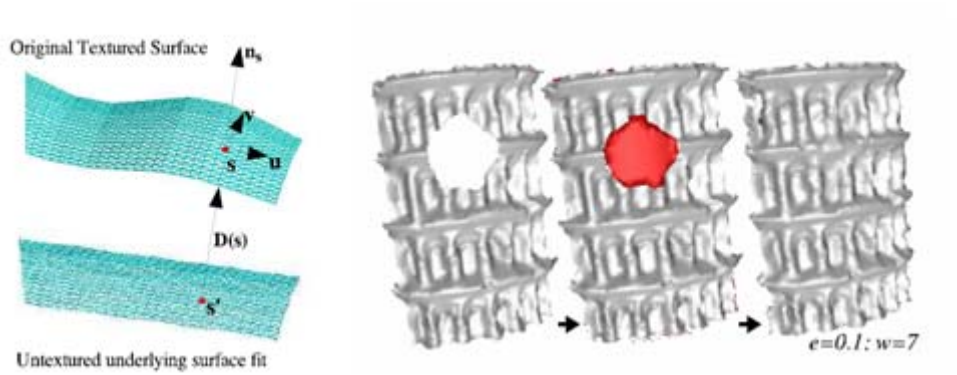

Figure 2.14: Breckon et al. [7] uses displacement vectors as the surface signature.

Park et al.[10] present a method to do surface completion that preserves both shape and appearance. They use grid to divide the surface into patches and do parameterization on each patch. For each patch, the six signatures of average, maximum and minimum of principle curvatures are used to perform shape similarity test. Although they use curvatures as signatures, the six statistic information of curvature can only represent the overall patch curvature but not the geometric context of the surface. In addition, by pasting the whole patch to a hole, the quality on the hole boundary can be noticeably different from other parts of the filling surface even with the use of surface blending.

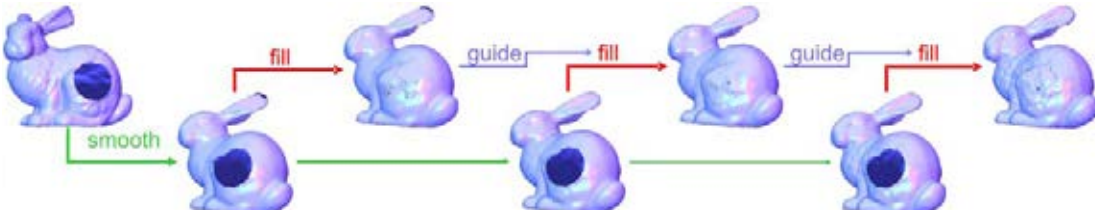

Figure 2.15: In iterative surface completion [9], The coarse filling is used as the guide for the more detail filling.

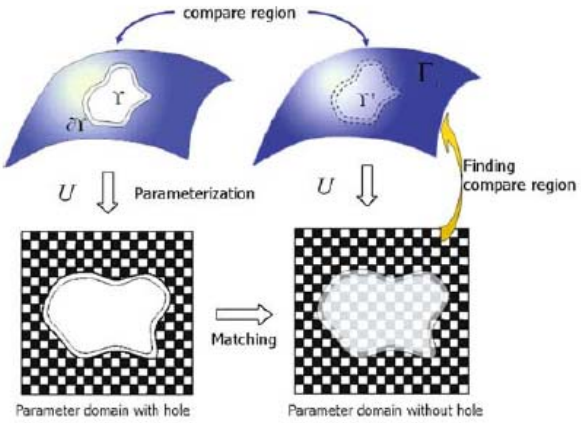

Figure 2.16: Park et al. [10] perform patches parameterizations and use patch-based synthesis framework on the planar surface.

All of the above methods have shown impressive results but do not truly use signatures that can reflect the local properties of the surface. Thus, especially for near-regular or regular relief patterns, it is arguable that the result surfaces from the similarity query are the best candidate to fill the hole. Furthermore, for a large hole, the filled area should also preserve the overall surface structure of the model. The example-based method that fill small patch by patch lacks the knowledge of the higher view of the surface structure.

2.2.3 Example-based Texture Synthesis

Example-based texture synthesis creates texture at any size without visible repetition by using the fixed-size texture as input. The example-based method can be broadly divided into two type of algorithm namely pixel-based and patch-based synthesis. In pixel-based synthesis [16-17], the value of each output pixel is determined by comparing its spatial neighborhood with all neighborhoods in the input texture. The input pixel with the most similar neighborhood will be assigned to the corresponding output pixel. In contrast, patch-based synthesis [37-40] iteratively copies the whole texture patch to the target area instead of a single pixel at a time. However, patch-based synthesis has to solve the problem of overlapped, possibly conflict, region when copying patches. Example-based techniques also used in other applications such as image inpainting [41-42] to fill the missing background image.

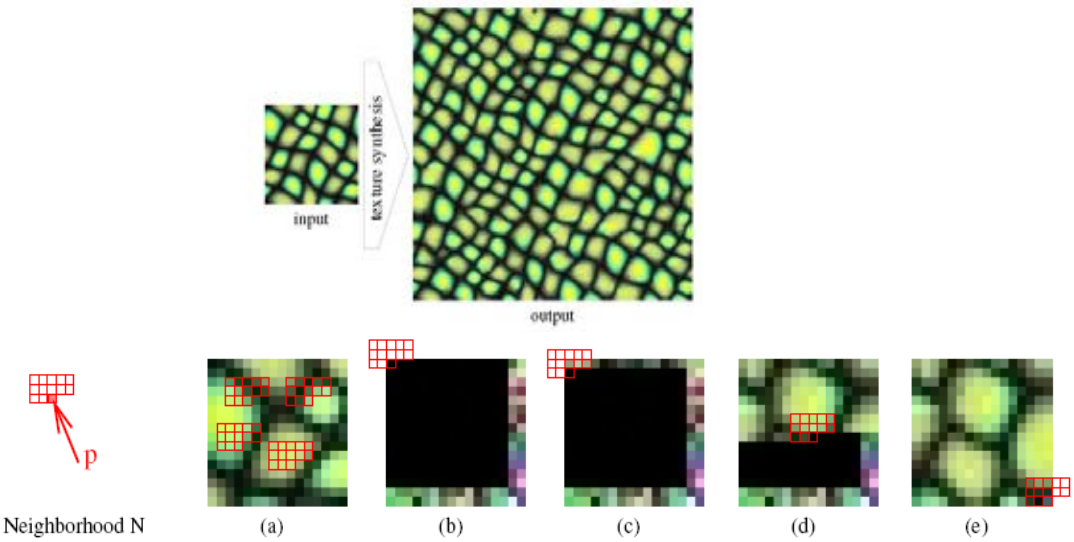

Figure 2.17: Pixel-based texture synthesis [43].

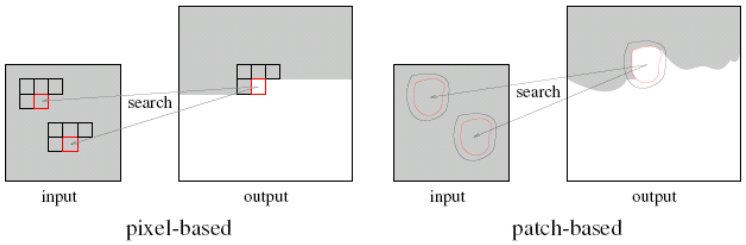

Figure 2.18: The comparison between pixel-based and patch-based method [43].

2.2.4 Example-based Texture Synthesis on Surfaces

Another group of works focus on synthesis texture on 3D surfaces. Texture synthesis directly on the surface is more challenging since it is not easy to specify the orientation and neighborhood information on surfaces. Approaches to tackle these problems are to densely populate surface with points and treat them like pixels [15], unfold the mesh onto the plane and perform synthesis in 2D [44] or treat triangles like patches, and find per-vertex (u, v) coordinates [45].

Figure 2.19: Texture synthesis directly on the surfaces [15].

This thesis uses example-based framework to transfer relief pattern to the smoothly filled hole. Orientation and neighborhood definition of each point on surfaces are also the problem that is needed to be addressed in the thesis. Densely populate points [15] looks promising but may be too burden to use in the mesh domain.

2.2.5 Geometric Synthesis

Geometric synthesis constructs novel surfaces based on example surfaces. Volumetric geometric creation [46] uses the detail surface from the given examples and places that detail to the target surface. Mesh quilting [47] uses example surface patches to stitch together over the surface of the target mesh. These approach synthesis new geometries by way of copy-and-paste method. This thesis takes a different approach by trying to extract the geometric detail from the given surface and fuse it with the smoothly filled hole.

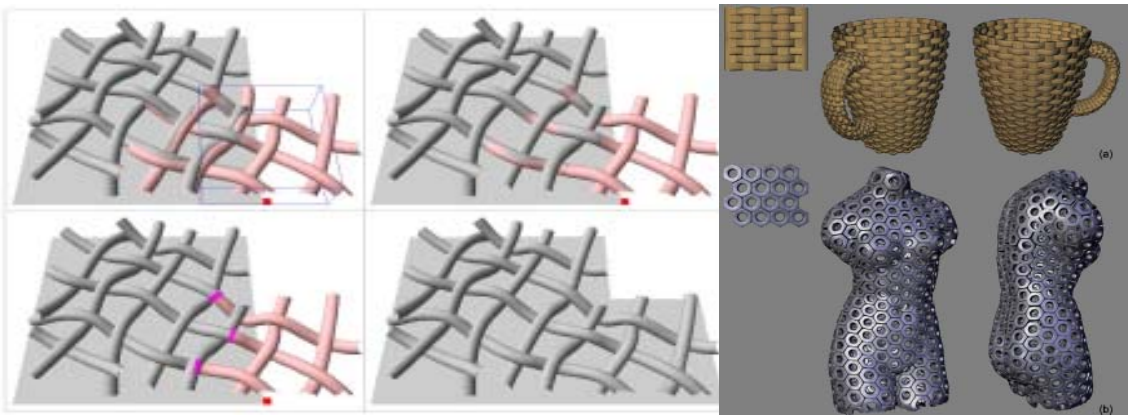

Figure 2.20: Mesh quilting stitches the exemplar meshes together and places them on the target surfaces [47].

2.2.6 Mesh smoothing

Mesh smoothing is a tool in geometric processing. The goals of mesh smoothing are denoising and generating fair surfaces. Surface denoising is the method to remove high frequency noise of the surface. Fair surfaces are the surfaces that satisfy some aesthetic requirement [48]. Usually, fair surfaces are defined by some energy function such as curvature function [36]. Mesh smoothing can be view as performing ideal low-pass filter on the surface [49].

Curvature flow [50] uses diffusion process to perform mesh smoothing. The area with high curvature is diffuse to the area with low curvature. It is a powerful and widely used tool to perform mesh smoothing [49].

Diffusion process of the function f is defined as linear diffusion equation

$$\frac{\partial f(x,t)}{\partial t} = \lambda \frac{\partial^2 f(x,t)}{\partial x^2} \tag{14}$$

where λ is the rate of diffusion.

In Curvature flow, where f is defined as surface x, the rate of change of the function over time should be proportional to the curvature of the surface. The right hand side of the equation is the Laplacian operator.

$$\frac{\partial x}{\partial t} = \lambda L(x) \tag{15}$$

The Laplacian operator can be described as

$$L(x_i) = \sum_{j \in N(x_i)} w_{ij}(x_j - x_i) \tag{16}$$

where x_i is the vertex position, $N(x_i)$ is the set of neighborhood vertices, w_{ij} is the weight. The weight can be defined as follows.

Scale-dependent weight [51] is defined as

$$w_{ij} = \frac{1}{|e_{ij}|} \tag{17}$$

where e_{ij} is the edge that connect vertex i and vertex j.

Curvature normal weight [50] is defined as

$$w_{ij} = \cot\alpha_j + \cot\beta_j \tag{18}$$

where α_j and β_j are the angles opposite the edge in the two triangles that share the edge e_{ij}. Curvature normal weight is a better approximation of Laplacian than Scale-dependent weight.

Diffusion process can sometime smooth out the existing features of the surfaces such as sharp edges and corners. Anisotropic diffusion [52] can alleviate these problems by perform diffusion anisotropically. Data dependent function is used to guide the flow of anisotropic diffusion. Bilateral filter [53-54] is feature-preserving filtering technique. The spatial distance and local variation of normals is taken into account while smoothing.

2.2.7 Mesh parameterization

Mesh parameterization is the method to compute one-to-one mapping between two surfaces with similar topology [55]. The squared planar surface can be parameterized by the parameter u and v which have the ranges of zero to one.

When mapping arbitrary 2-mainfold surface with planar surface, the distortion is unavoidable. The mapping can cause distortion in length, angle and area. The parameterization algorithms are designed to minimize the distortion as much as possible.

Let $p=(x,y,z)$ be the points of a given surface and let $\bar{u}=(u,v)$ be the parameter points of squared planar surface. The goal is to find the mapping $g, \bar{u}_i = g(p_i)$.

First, for boundary vertices $p_i \in V_B$ theirs corresponding parameter points $\bar{u}_i$ are set as the boundary values. Usually, one of the components of parameter points has the value of zero.

This research uses affine combination method or barycentric map to find the mapping g. In this approach, each interior parameter point is an affine combination of its neighbors [55-56].

$$\bar{u}_i = \sum_{j \in N_i} \lambda_{ij} \bar{u}_j \tag{19}$$

$$\sum_{j \in N_i} \lambda_{ij} = 1 \tag{20}$$

λ_{ij} is called barycentric coordinates of p_i which can be formulated by the normalization.

$$\lambda_{ij} = \frac{\omega_{ij}}{\sum_{k \in N_i} \omega_{ik}} \tag{21}$$

Using the Mean value coordinates [57], ω_{ij} can be defined as

$$\omega_{ij} = \frac{\tan\frac{\alpha_{ij}}{2} + \tan\frac{\beta_{ij}}{2}}{r_{ij}} \tag{22}$$

The notations in of the above equation are described in Figure 2.21.

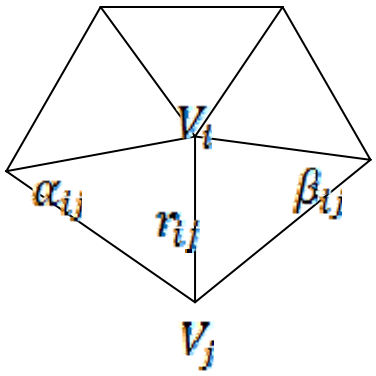

Figure 2.21: The notations of angles and lengths used in Mean value coordinates.

The linear system of barycentric coordinates can be over-determined. Therefore, least-square method is used to find the solution for the system.

The fixing of boundary may not need to be square. The free-boundary parameterization [58] can give less distorted mappings. The shape of the boundary is not limited to convex shape. These methods perform distortion analysis on the given surfaces and then minimize the distortion.

Planar parameterization can only handle surface with disk topology. Thus for models with high genus, segmentation are performs on the models first [59-60]. Each part that has disk shape topology is then parameterized individually.

CHAPTER III

METHODS

3.1 Overviews

This work represents the 3D surfaces with polygon meshes. The input of the algorithm is a mesh with a hole. The mesh must be regular, uniform and composed of isotropic triangles. The fairing method can be applied if the input mesh does not have the specific qualities. If the model contains many holes, each hole is filled independently, one hole at a time. The overview of the algorithm is shown in Figure 3.1 and its corresponding pseudocode in Figure 3.2.

The process of the algorithm is to smoothly fill the hole first and then extract the relief pattern from the surrounding surface and transfer it to the filled hole. The challenges of this work are on how to extract the relief pattern, how to represent it and how to transfer it to the smoothly filled hole.

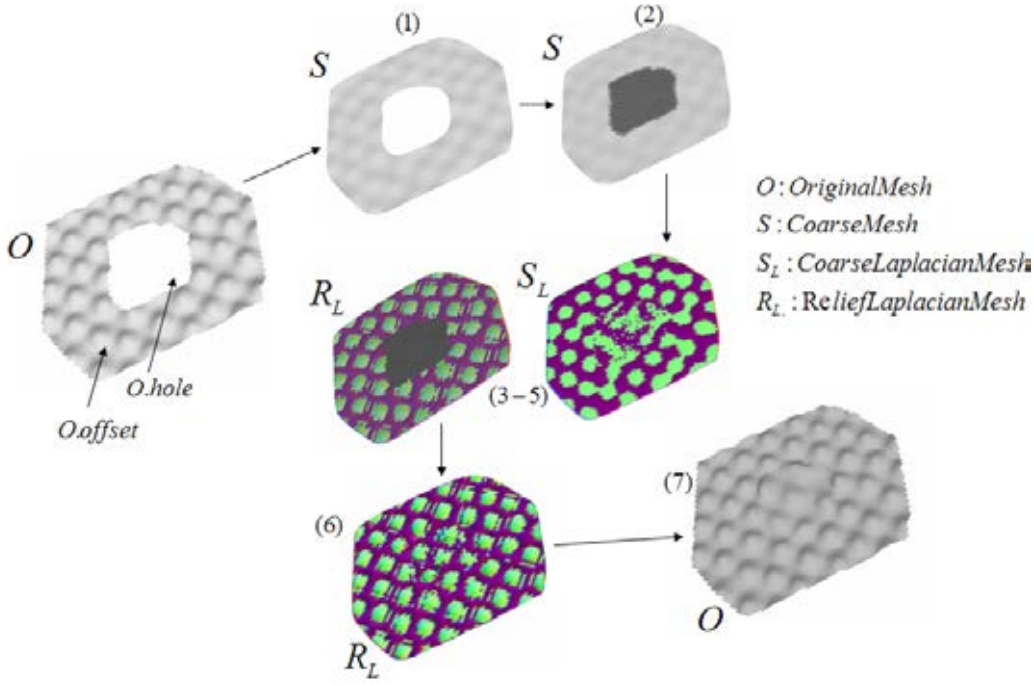

Figure 3.1: The high-level algorithm pipeline.

As discussed in the previous works, the example-based framework has proven itself to be successful in filling textures. It looks promising to use this framework for transferring relief pattern. However, in order for example-based method to be successful, it needs a signature that reflect the pattern and a signature that is defined spatially on the given exemplar.

The key idea of the algorithm is to use multi-resolution decomposition to extract the relief pattern and to use Laplacian coordinate to represent it. Laplacian coordinate defines the local geometric properties, which are normal and curvature, of each surface point. The surface signature can be defined for each vertex using the Laplacian coordinate of neighborhood vertices. In this way, this Laplacian signature can be used with the example-based framework to transfer the relief pattern to the smoothly filled surface.

High-level Algorithm

1) ***S*** = Mesh_Smooth(***O***)
2) Smooth_Fill_Hole(***S***)
3) ***SL*** = Laplacian_Transform(***S***)
4) ***OL*** = Laplacian_Transform(***O***)
5) ***RL.offset*** = ***OL.offset*** – ***SL.offset***
6) ***RL.hole*** = Fill_Laplacian(***RL.offset***)
7) ***O*** = Inverse_Lap(***RL*** + ***SL)***

Figure 3.2: Pseudocode of the algorithm.

For an input surface O, the offset region around the hole boundary is computed to use as a relief pattern exemplar. The offset region should have the area at least four to five time larger than the hole area. Specifically, the region must cover the relief pattern.

In computing the offset region, user defines the offset radius. The offset radius is the distance from the hole boundary to the offset boundary. The algorithm starts by setting the distance of each vertex on the hole boundary to be the value of

zero. Then, Dijkstra's algorithm is used to assign the distance to the others vertices that are connected with the hole boundary vertices. The distance is the accumulated edge length measured from the hole boundary. The assigned vertices are tagged as visited and are put into queue. The algorithm dequeues those vertices and explores the unvisited vertices in the breadth first search manner until the offset radius distance is reached. Polygons which contain vertices that have the distances less than the offset radius are in the offset region. The other polygons outside the offset region are not used in the computation. The edges that are shared by the polygons of the offset region and the polygons outside the offset region are used as the offset boundary.

Mesh O is divided into two regions: region $O.hole$ which is the region that is needed to be filled and is now empty and region $O.offset$ which is the offset region around the hole boundary (Figure 3.1). Other meshes that are constructed further in the algorithm are composed of the hole part and the offset part. The surface part outside the offset region is not use in the surface completion algorithm. Meshes with subscript L are represented in Laplacian coordinate. Those that do not are represented in Cartesian coordinate.

The algorithm performs mesh smoothing on mesh O and obtain the coarse mesh S as the result (step1, in Figure 3.2). This mesh, S, is smoothly filled (step 2, in Figure 3.2). Mesh O and mesh S are transformed into Laplacian coordinate using Geometric Mesh Laplacian which result in mesh O_L and mesh S_L respectively (step 3-4, in Figure 3.2). Mesh S_L and mesh O_L have the same mesh topology as in mesh S and mesh O but instead of storing vertex position, they store Laplacian coordinate.

The relief mesh, R_L, is obtained by subtracting O_L with S_L (step 5, in Figure 3.2). The hole's region of the relief mesh, $R_{L.}.hole$, is filled using the relief example from the offset region, $R_{L.}.offset$ (step 6, in Figure 3.2). Finally, the coarse mesh $S_{L.}.hole$ and relief mesh $R_{L.}.hole$ are combine to reconstruct the completely filled mesh $O.hole$ (step 7, in Figure 3.2).

3.2 Multi-resolution Decomposition on Mesh

Multi-resolution decomposition views surface as compose of low-frequency and high-frequency information. The low-frequency part is interpreted as the coarse mesh and high-frequency part as the relief mesh. The coarse mesh is the overall surface structure while the relief mesh contains the geometric detail of the surface. Each level of detail is filled separately using different algorithm. The filled surfaces of each resolution are then combined together to form the final filled surface. In this way, the filled surface can preserve both the surface structure and surface contextual detail.

Usually, the coarse mesh is obtained by perform mesh smoothing and the relief mesh is computed using normal displacement method. Normal displacement [61-62] uses the normal of the base surface or coarse surface to sample the original surface in order to get the displacement vectors. The normal displacement can be written as

$$p_i = b_i + h_i \cdot n_i \tag{23}$$

where p_i is the vertex position of the original surface, b_i is the vertex position of the base surface, n_i is the normal vector of that vertex and h_i is the displacement value in the direction of n_i and is used to represent relief information. (see Figure 3.3)

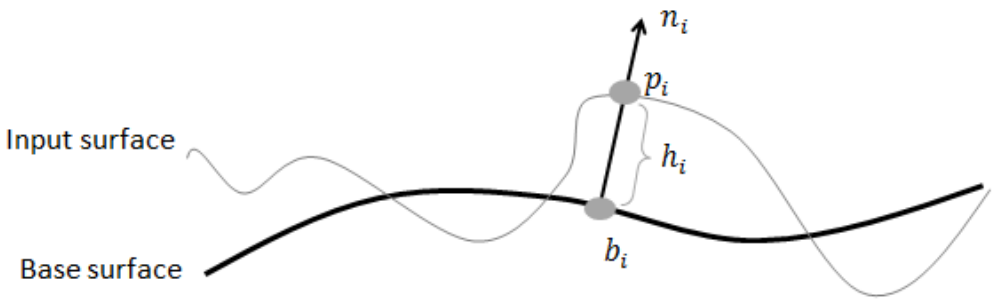

Figure 3.3: Relief extraction using Normal displacement method.

The problem of this method is that the surface has to be representable by a height field function. Otherwise, the relief mesh does not have a one-to-one

mapping with the base mesh. Depend on the geometry of the base mesh, sampling the original mesh can introduce distortion in geometric information. The distortion is high at the steep area of the surface. Due to undersampling artifact, the neighbor displacement vectors can have discontinuity even though the original surface is connected.

In this research, Laplacian representation is proposed to use with the multi-resolution decomposition. Laplacian coordinate does not suffer from the sampling problem as in the Normal displacement method and can be used to represent any 2-manifold surfaces. The high frequency detail is represented as the difference between the Laplacian coordinate of the original surface and the Laplacian coordinate of the coarse detail surface (step 3-5, in Figure 3.2).

$$R_L = O_L - S_L \tag{24}$$

As stated in the chapter 2, Laplacian coordinate is equal to mean curvature normal.

$$\Delta_S x = -2Hn \tag{25}$$

Actually, Laplacian coordinate is an ambiguous representation of the mean curvature normal. The mean curvature, H, is positive when the shape in convex and negative when the shape in concave. Thus, for the two points that have opposite normal vector direction and different mean curvature sign are equal in term of Laplacian representation.

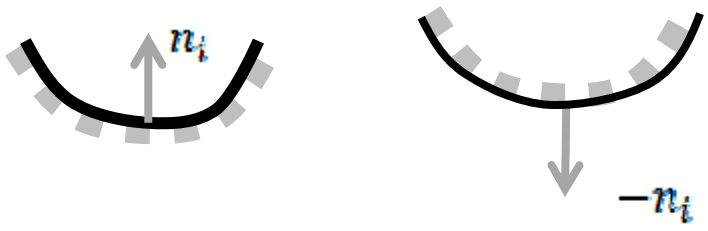

Figure 3.4: An example which Laplacian representation can be ambiguous.

In representing the relief mesh, R_L cannot be used directly as the relief mesh since its Laplacian coordinate, δ, is still in the world coordinate. The Laplacian of the relief mesh has to be defined relatively to the coarse mesh. Since, this research views original mesh as compose of relief mesh and coarse mesh. The Laplacian coordinate, δ, of each vertex of R_L has to be transformed to tangent space of the corresponding vertex of the coarse mesh. For each vertex, the three axis of the tangent space can be defined using the normal vector, the binormal vector and the tangent vector of that vertex. The normal vector of the coarse mesh is readily available. However, mesh parameterization is needed to compute the binormal vector and the tangent vector.

For a given vertex on the coarse mesh S, let B, N, T be the normal vector, tangent vector and binormal vector of that vertex respectively,

The matrix $T_{W \to T}$ that used to transform Laplacian coordinate from world space to tangent space can be defined as

$$T_{W \to T} = \begin{bmatrix} B_x & B_y & B_z \\ N_x & N_y & N_z \\ T_x & T_y & T_z \end{bmatrix}$$

The transformation can be compute as follow.

$$\begin{bmatrix} B_x & B_y & B_z \\ N_x & N_y & N_z \\ T_x & T_y & T_z \end{bmatrix} \begin{bmatrix} \delta_{world}^{\ x} \\ \delta_{world}^{\ y} \\ \delta_{world}^{\ z} \end{bmatrix} = \begin{bmatrix} \delta_{\tan gent}^{\ x} \\ \delta_{\tan gent}^{\ y} \\ \delta_{\tan gent}^{\ z} \end{bmatrix} \quad (26)$$

R_L is now ready to be used to represent the relief mesh. The ambiguous on mean curvature normal representation can now be resolved. In tangent space, if the δ^y component is negative then the mean curvature is negative (concave) and vice versa.

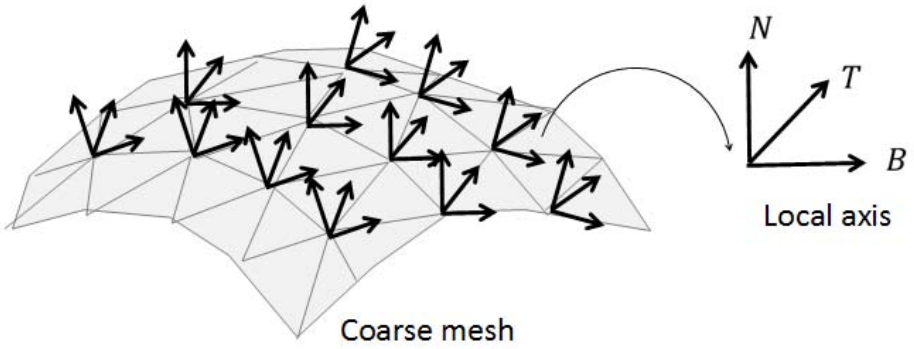

Figure 3.5: The local axis is defined for each vertex of the coarse mesh.

Figure 3.6 (b, e) shows the normal component of the Laplacian representations. The green color represents the normal vectors that have the same direction as the normal vectors of the coarse mesh, while the blue color represents the normal vectors that have the opposite direction with the normal vectors of the coarse mesh. Figure 3.6 (c, f) shows the mean curvature component of the Laplacian representations. The blue color indicates the positive curvature. The red color indicates the negative curvature. The green color indicates the zero curvature.

Laplacian coordinate is used in the relief mesh completion phase as the shape signature since it can handle any 2-manifold surfaces and reflects the surface geometrical information. It also does not introduce sampling artifact such as discontinuity, distortion and non-uniform sampling. The definition of the Laplacian surface signature to use with example-based framework is discussed in section 3.5.1.

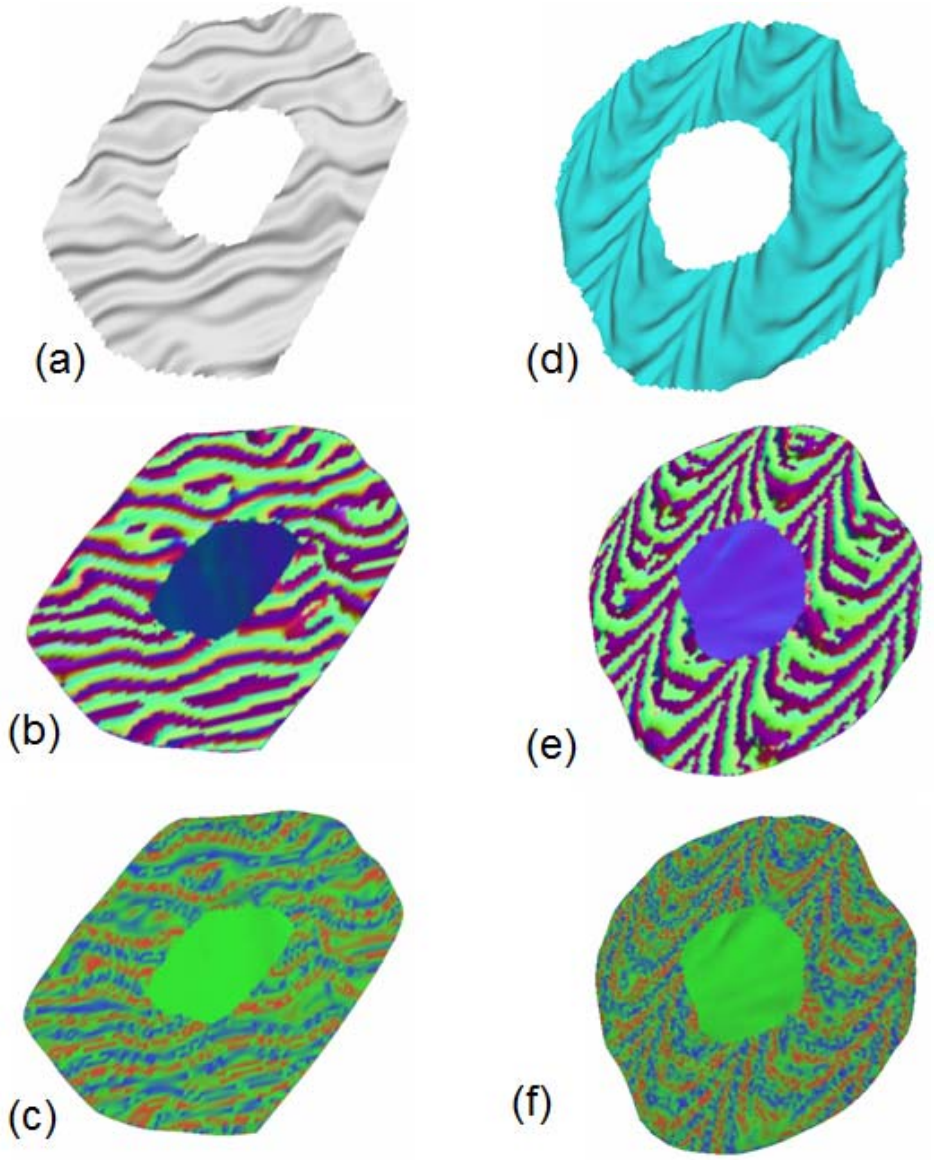

Figure 3.6: The visualization of the Laplacian representation.

3.3 Coarse Mesh Completion

The coarse mesh $S.offset$ is obtained by performing mesh smoothing. This research uses Curvature flow to do mesh smoothing [50] on $O.offset$.

This research uses Curvature normal weight on interior vertices and uses Scale-dependent weight for hole boundary vertices, since Curvature normal weight cannot be defined for boundary vertices.

In addition, the neighborhood vertices used to compute Scale-dependent weight are limited to vertices that are on the hole boundary. Otherwise, the boundary

vertices will be heavily pulled by the interior vertices due to the imbalance distribution of the neighborhood vertices.

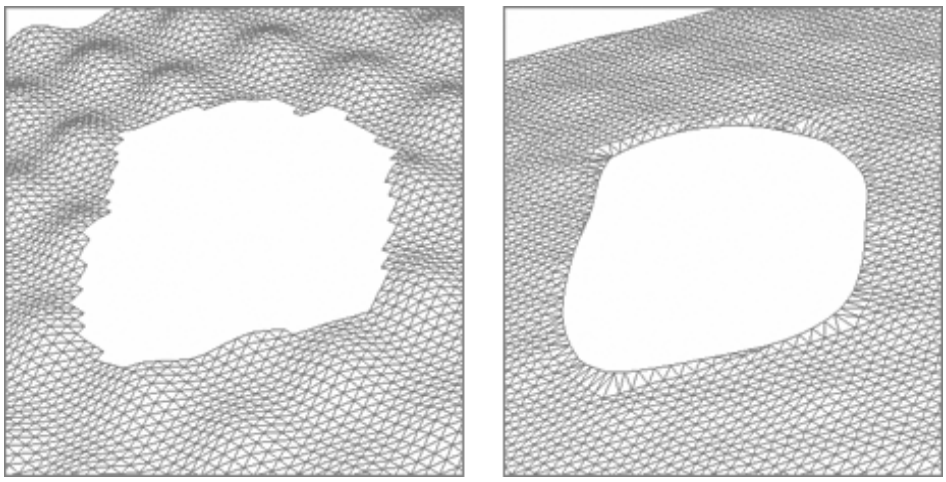

Figure 3.7: The hole boundary is rounder due to effect of mesh smoothing.

Explicit Euler integration method or Finite differences method is used to solve the differential equation. This can be computed by the iterative algorithm using the update rule

$$P_{new} \leftarrow P_{old} + \lambda L(P_{old}) \tag{27}$$

where P is the vertex position. The update rule is computed for each x, y, z coordinate separately.

Mesh smoothing alternates the shape of the hole boundary considerably (Figure 3.7). This seems to be problematic at first as the shape of the filled hole is not match the shape of the original hole boundary. However, with the use of Laplacian representation, $O_L.hole$ can be reconstructed by choosing the anchor vertices to be the positions of the original hole boundary. The reconstructed $O.hole$ fuses seamlessly with the $O.offset$. The least-square solver attempts to reconstruct the Cartesian coordinate while preserve the mean curvature normal of each point as much as possible. The geometric distortion that may happen is distributed over the surface and hence unnoticeable. For Laplacian reconstruction, altering the hole boundary or keep the boundary intact makes no difference, since the anchor points needed to be set anyway.

On the other hand, if the hole boundary is fix when performing mesh smoothing (Figure 3.8), $S.hole$ computed from the coarse mesh completion will not be smooth and it will not reflect the coarse resolution of the original surface. The result will have a downside effect on the relief mesh extraction.

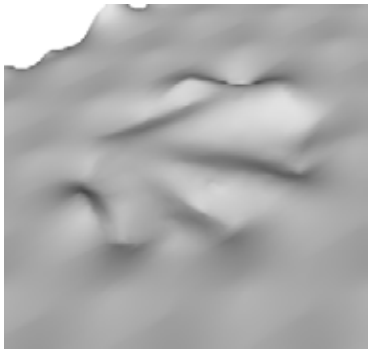

Figure 3.8: Fixing the hole boundary produces bumpy coarse mesh on the hole region.

From the mesh smoothing results in Figure 3.7, the faces around the hole boundary are lager than the interior faces, because Scale-dependent weight only uses boundary vertices as the neighborhood. This problem can be alleviated by performing tangential relaxation on the coarse mesh $S.offset$ (Figure 3.9). Tangential relaxation is the method to make the faces more uniform by repositioning vertices. In tangential relaxation, the vertices can move only in the direction of its tangent vectors. This constrain preserves the direction of normal vectors of vertices.

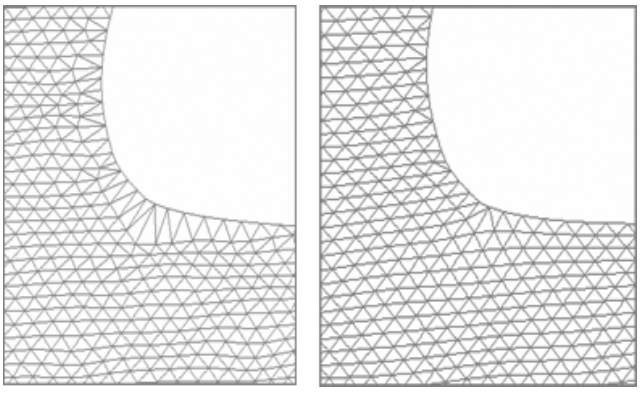

Figure 3.9: Tangential relaxation adjusts the areas of the polygons to make them more uniform.

This research use Liepa's algorithm [1] to perform coarse mesh completion. The algorithm performs hole triangulation, mesh refinement and fairing. Liepa's algorithm is used in coarse mesh completion because it can produce smooth filling surface with minimize surface area and surface dihedral angle.

In the fairing step, the Bi-Laplacian equation is needed to be solved.

$$\Delta^2 x = 0 \tag{28}$$

In this research, the discretization method is used to replace differential operator with the divided different operator.

$$\Delta x = U(x_i) = \frac{1}{n} \sum_{j \in N(x_i)} (x_j - x_i) \tag{29}$$

$$\Delta^2 x = U^2(x_i) = \frac{1}{n} \sum_{j \in N(x_i)} (U(x_j) - U(x_i)) \tag{30}$$

where $N(i)$ are the neighborhood vertex of i. n is the number of neighborhood vertex of i.

The Bi-Laplacian system is solved using forward Euler integration methods. The region $S.hole$ is now available.

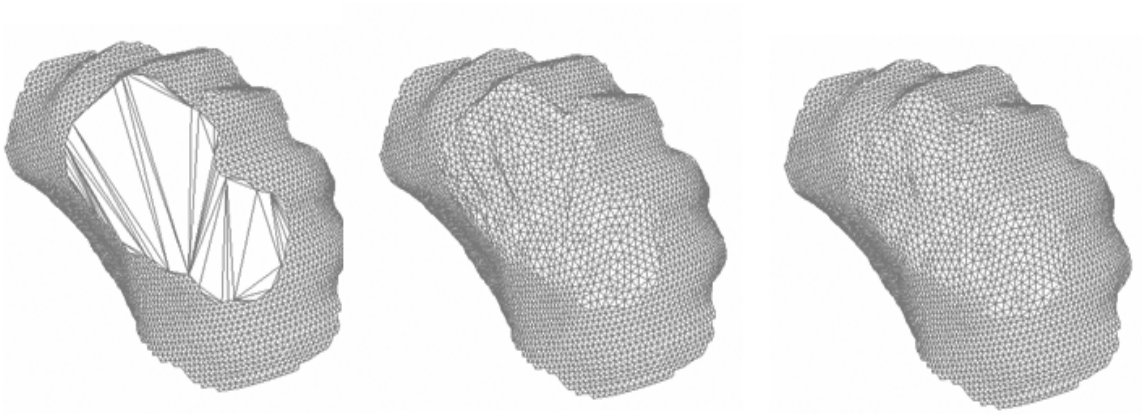

Figure 3.10: Smooth surface completion using Liepa's method, (left) hole triangulation, (middle) mesh refinement, (right) fairing.

3.4 Computing Mesh Parameterization

As introduced in section 3.2, mesh parameterization is needed to defined binormal vectors and tangent vectors for computing $R_{L,\tan gent}$. The Laplacian signature that is used with the example-based framework also requires surface parameterization to consistently defined the orientation of each surface point.

This research computes the mapping between squared planar surface and the surface of mesh O. This mapping can also be used with mesh S and mesh R, since both have the same mesh topology as mesh O.

This research uses barycentric map to construct parameterization. The mapping is then used to define the binormal vectors and tangent vectors for each vertex points on the coarse mesh. The result parameterization of barycentric map technique is close to the conformal mapping, the mapping which preserves angle distortion. Thus, conformal mapping can consistently define the local axis over the surface. The distortion in length can be negligible in computing local axis.

In Figure 3.11, (a-c) are the input surfaces and (d-f) are the result parameterizations respectively. The results in Figure 3.11 have high stretching near the boundary region. This is because the solver tries to preserve the angles of vertices as much as possible. In the image (g) of Figure 3.11, the u, v parameter [0,0] and [1,1] are mapped with the corner of the square mesh. The result parameterization has little distortion. In contrast with the image (h), the u, v parameter [0,0] and [1,1] are mapped at the center of the edge. This setting is the hard case for the mapping and causes highly stretching in the mapping. However, in the center region, the angle is still finely preserved.

Although there is some distortion in the mapping, this is not going to be an issue, since this research only uses parameterization for defining the binormal direction and the tangent direction for each point on the surface.

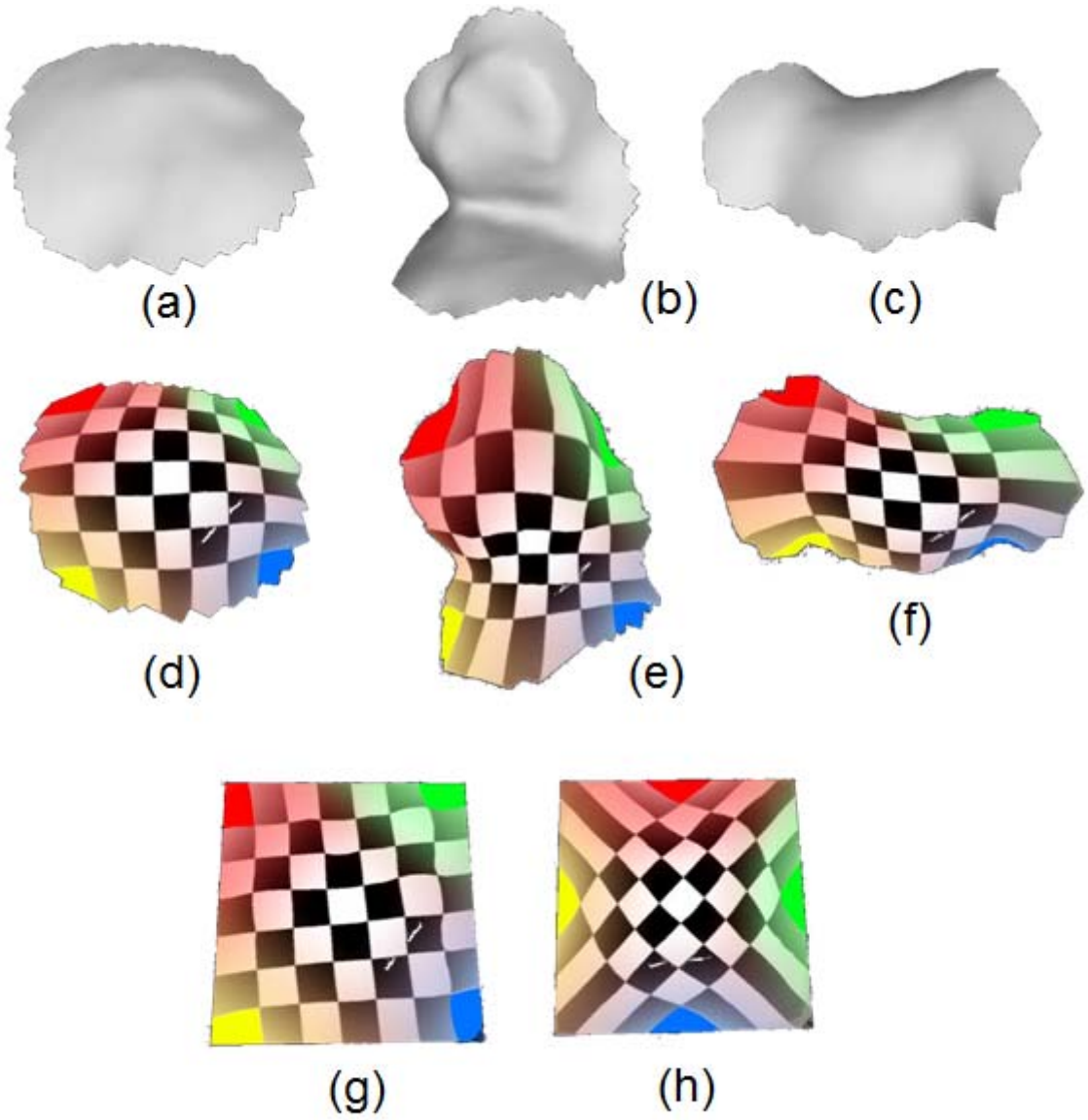

Figure 3.11: The results of mesh parameterization.

3.5 Relief Mesh Completion

The example-based framework is used to fill the region $R_{L.}.hole$. The region $R_{L.}.offset$ is used as the exemplar. The relief information of $R_{L.}.hole$ is still empty since the hole region is only smoothly fill and have no relief information. The Laplacian coordinates of every vertices in $R_{L.}.hole$ have value of zero. The relief pattern encoded in Laplacian coordinate from the offset region are used to transferred to $R_{L.}.hole$. The pseudocode is presented in Figure 3.12.

This research uses pixel-based synthesis [16] approach by transferring the Laplacian coordinate of the hole region to the offset region vertex by vertex. The patch-based synthesis can have an issue with the differences in mesh topology between $R_{L.}.hole$ and $R_{L.}.offset$. If the mesh topologies of the two regions are different, Laplacian coordinates cannot be copied directly from one region to the other. In addition, patch-based synthesis can complicate the surface signature definition since the shape and the area of each patch may vary from each other.

Relief mesh completion

1) For each vertex V_o in $R_{L.}.offset$
2) compute its Laplacian signature $\xi(V_o)$
3) For each vertex V_h in $R_{L.}.hole$
4) compute its Laplacian signature $\xi(V_h)$
5) For each V_o of $R_{L.}.offset$
6) *dist* = compare_distance($\xi(V_o)$,$\xi(V_h)$)
7) if (*dist* < *min_dist_sofar*)
8) j = index_of(V_o)
9) $\delta_h = T_{T \to W} \delta_j$

Figure 3.12: The pesudocode for relief mesh completion.

Let V_O be the set of vertices of $R_{L.}.offset$, V_o is an element in V_O.

Let V_H be the set of vertices of $R_{L.}.hole$, V_h is an element in V_H.

For each vertex in $R_{L.}.offset$, V_o , its Laplacian signature $\xi(V_o)$ is computed from the collection of Laplacian coordinates of the sampling points in the neighborhood region of vertex V_o. (section 3.5.1).

The algorithm starts at the vertex on the boundary of $R_{L.}.hole$. It then visits other vertices of V_H in spiral fashion. The last visited vertex is the one located at the center of the hole region. The vertex visiting is implemented using queue and breadth first search algorithm. By visiting vertices in this manner, many neighborhood vertices around V_h will already have the Laplacian coordinates assigned and can be used to compute the Laplacian signature for the vertex V_h. If there are too few neighborhood vertices to use in computing the Laplacian signature, the Laplacian signature may not reflect the characteristic of the relief pattern of a surface patch.

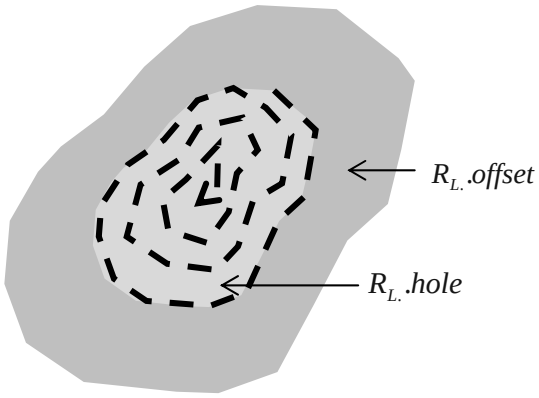

Figure 3.13: The vertices in $R_{L.}.hole$ are visited in spiral fashion, started from the hole boundary.

For each vertex in $R_{L.}.hole$, V_h, the algorithm searches for the V_o that have the best matching Laplacian signature (section 3.5.2). The Laplacian coordinate of V_o is transferred to V_h using matrix $T_{T\to W}$ (section 3.5.3).

The algorithm terminates when all vertices in V_H are visited and have received the Laplacian coordinate from V_O.

3.5.1 Computing Laplacian Signatures

The surface signature is used to compare the visual similarity between two patches of surface that are equivalent in surface areas. In this research, the surface signature that can be defined for every vertex on the surface is needed to use with the example-based framework.

Defining a signature on surface introduces many problems that are not found in texture. Vertices may not have an equal number of neighborhoods. Vertices may not uniformly distribute on the surface. These are the problems of irregularity of mesh topologies. Furthermore, surface is non-planar like texture.

In this research, for any given vertex, V_i, Laplacian surface signature $\xi(V_i)$ is defined as the collection of tangent space Laplacian coordinates δ_{1-n} of the sampling points in the neighborhood region of vertex V_i.

$$\xi(V_i) = \{\delta_1, \delta_2, ..., \delta_n\} \tag{31}$$

where n is the number of sampling point.

Laplacian coordinate is used because it can be defined on each point of the surface. It also represents the geometric properties of each point, the curvature and normal, which define the visual characteristic of a surface.

The neighborhood region of vertex V_i, $N(V_i)$, is a squared region center at V_i. The dimension of this region is discretized so that it is defined by the odd integer values, such as 5x5, 7x7. The dimensional unit is defined to be the inverse of the square root of the number of vertices in the mesh. Thus, for a mesh with fairly uniform polygons areas, the dimensional unit is approximately equal to the average edge length of a mesh.

The sampling points are sampled uniformly in grid-based style around a given vertex V_i with the step size equal to the dimensional unit. For example, the 7x7 region has 48 sampling points or 48 components (Vertex V_i is not used as the sampling point). The Laplacian coordinate of each sampling point is interpolated from the Laplacian coordinate of the k-nearest vertices around that sampling point. The k-nearest vertices are weighted according to the inverse squared distance of the sampling point.

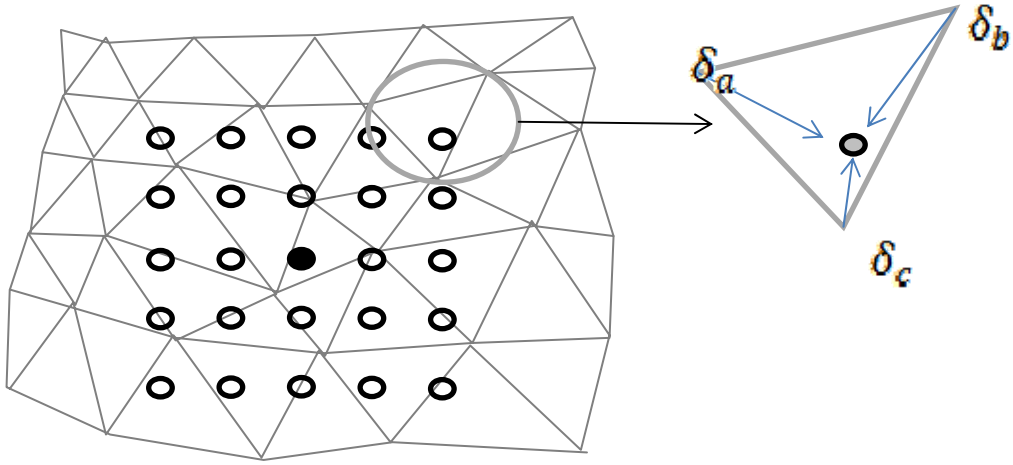

Figure 3.14: The 5x5 sampling points with 3-nearest neighbors.

In averaging the Laplacian coordinates, the mean curvature component, H, and the normal vector component, n, of the Laplacian coordinate are averaged separately. Before averaging normal vector, if δ^y is negative, it is inverted. This is needed due to the ambiguity of the Laplacian representation. The average mean curvature is then multiplied to the average normal vector.

Using Laplacian coordinates from the sampling points instead of vertices can solve the irregularity problem of mesh topologies. The sampling is also done directly on the surface, thus there is no sampling artifact as presented in normal displacement sampling.

In the regions around the mesh boundary, some sampling points may not be defined because they are out of bound and will not be included in the signature. In addition, sampling points around the unvisited V_h are not included in the signature since they do not have the relief information from the offset region. Thus, Laplacian surface signature of each vertex V_h can have different number of components.

The size of this region can be set by the user and uses for every vertex on the mesh. In practice, the neighborhood size should larger than the geometric feature of the surface, so that it can contain enough information of the relief pattern.

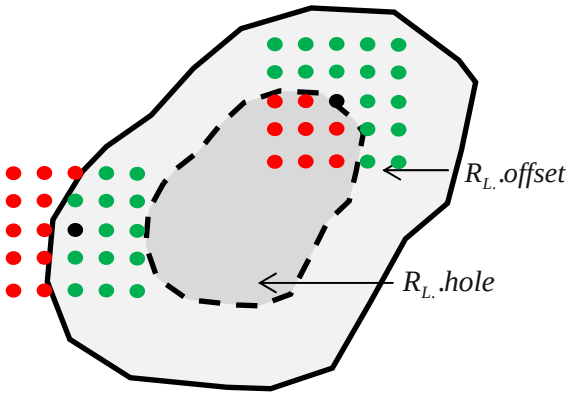

Figure 3.15: The red sampling points which are in the hole region or outside the offset region are not used to compute for the surface signature. Only the green sampling points are used to compute for the surface signature.

3.5.2 Comparing Laplacian Signatures

Two Laplacian signatures are comparable if they have the same number of components. The comparison is done to determine how the two patches are similar visually to each other. Two components from the different signatures with the same index are compared to each other one-to-one.

This research defines three types of distance metric to compare two Laplacian signatures. Laplacian coordinate represents mean curvature, H, multiply by unit normal vector $\vec{n}$, $\delta = H * \vec{n}$.

Given two Laplacian signatures

$$\xi(V_i) = \{\delta_{ik}\}; k = [1..n] \text{ and } \xi(V_j) = \{\delta_{jk}\}; k = [1..n]$$

where n is the number of sampling points in neighborhood region.

The first metric considers only the normal vector component of the Laplacian coordinate. The normal distance metric is defined as

$$Dist_{\vec{n}} = \frac{1}{n}\sum_{k=1}^{n}\left(\left(\vec{n}_{ik}^{\,x} - \vec{n}_{jk}^{\,x}\right)^2 + \left(\vec{n}_{ik}^{\,y} - \vec{n}_{jk}^{\,y}\right)^2 + \left(\vec{n}_{ik}^{\,z} - \vec{n}_{jk}^{\,z}\right)^2\right) \quad (32)$$

The second metric considers only the mean curvature component of the Laplacian coordinate and is defined as

$$Dist_H = \frac{1}{n}\sum_{k=1}^{n}\left|H_{ik} - H_{jk}\right| \quad (33)$$

The third metric considers both components of the Laplacian coordinate and is defined as

$$Dist_{H\vec{n}} = w \ * Dist_{\vec{n}} + (1-w) * Dist_H \quad (34)$$

where w is the weight.

Curvature and normal component are independent to each other. Thus, they can be compared separately.

This research uses uniform weight for all components in the signature. The far-off sampling points from V_i will receive the same weight as the sampling points near V_i. Each sampling point is equally important in determine the visual similarity of a surface patch. If, however, the weight is assigned differently according to the radius from V_i, the result comparison will be the same as comparing two signatures with smaller neighborhood size.

Finding the best matching $\xi(V_j)$ for all given $\xi(V_i)$ of the hole region is done in a brute force style thus result in quadratic time complexity. This part can be speed up by using acceleration structure such as k-d tree to store all the signatures of the offset region and then use this tree to compare with the signature from the hole region. This adaptation can result in $O(n \log n)$ complexity. However, the comparison may not be done fairly for each component of the signature. The components near the root node would have more influence than the components near the leaf nodes.

3.5.3 Transferring Laplacian coordinate

When transferring the Laplacian coordinate of V_o to V_h , the different in transformed space has to be considered. Laplacian coordinate of V_o is in tangent space. However, Laplacian coordinate in world space is needed in the Cartesian coordinate reconstruction stage (section 3.6).

For a given vertex on the coarse mesh $S.hole$, let B, N, T be the normal vector, tangent vector and binormal vector of that vertex respectively,

The matrix $T_{T\to W}$ that used to transform Laplacian coordinate from tangent space to world space can be defined as

$$T_{T\to W} = \begin{bmatrix} B_x & N_x & T_x \\ B_y & N_y & T_y \\ B_z & N_z & T_z \end{bmatrix}$$

The transformation can be compute as follow.

$$\begin{bmatrix} B_x & N_x & T_x \\ B_y & N_y & T_y \\ B_z & N_z & T_z \end{bmatrix} \begin{bmatrix} \delta_{\text{tan}\,gent}^{\;x} \\ \delta_{\text{tan}\,gent}^{\;y} \\ \delta_{\text{tan}\,gent}^{\;z} \end{bmatrix} = \begin{bmatrix} \delta_{world}^{\;x} \\ \delta_{world}^{\;y} \\ \delta_{world}^{\;z} \end{bmatrix} \tag{35}$$

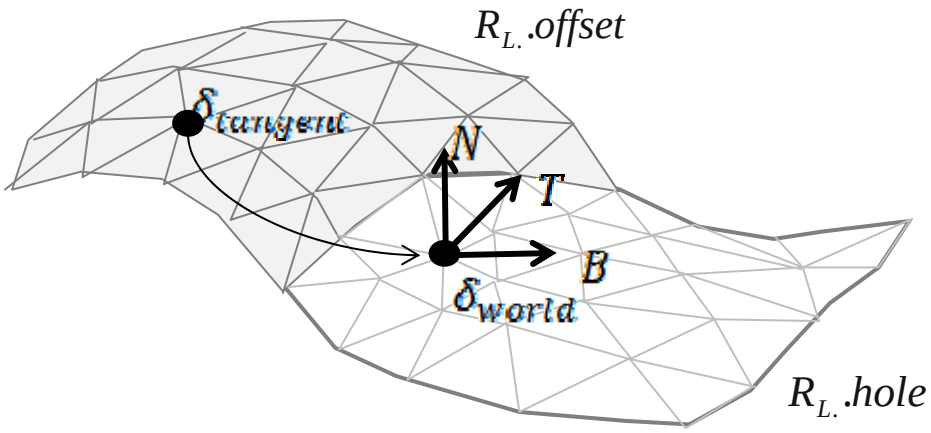

Figure 3.16: When transferring Laplacian coordinate, a transformation to world space is required.

3.6 Combining Coarse Mesh and Relief Mesh

The Laplacian coordinate of mesh $R_{L.}.hole$ and $S_{L.}.hole$ are combined to obtain $O_{L.}.hole$. Mesh $O.hole$ is reconstructed from mesh $O_{L.}.hole$ using inverse Laplacian transformation. The boundary vertices are used as anchor points. The corresponding Cartesian coordinate of the boundary vertices of mesh O are assigned to these anchor points. Thus, the algorithm does not alter the original surface at all. This feature can be important for some applications that have to maintain the original surface data.

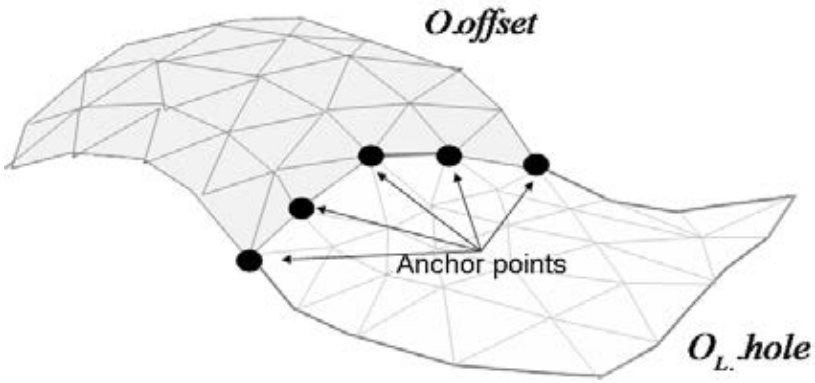

Figure 3.17: Vertices on the hole boundary are used as the anchor points.

The curvature of mesh $R_{L.}.hole$ and $S_{L.}.hole$ are blended smoothly in the Laplacian coordinate space. In addition, reconstruction by least-square minimization method distributes the distortion error all over the filling surface hence make the error (from minimization) hardly be noticeable. This is in contrast with the Normal displacement method which the discontinuity of displacement vectors cannot be blended easily in Cartesian coordinate.

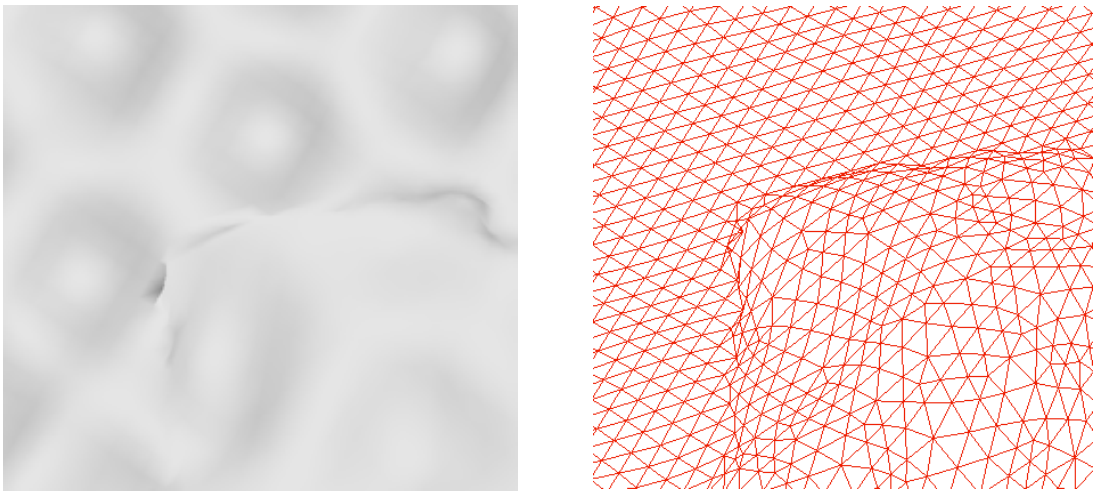

Figure 3.18: The triangles around the hole boundary are heavily pulled by the anchor points in the Inverse Laplacian transform.

Triangles around the boundary of $O.hole$ can be compressed because of the anchor points constrain (Figure 3.18). This problem can be solved by performing tangential relaxation on mesh $O.hole$ (Figure 3.19).

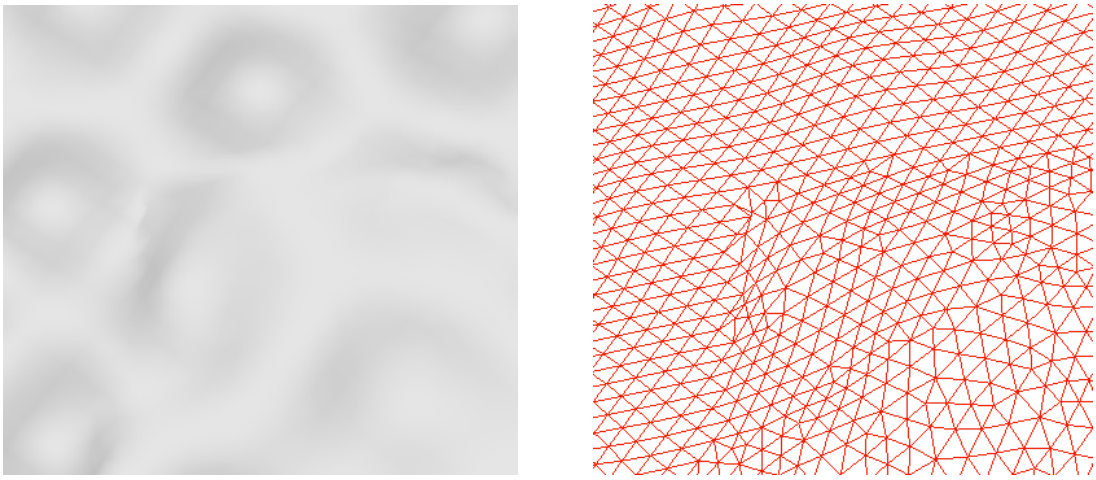

Figure 3.19: Tangential relaxation is used to solve the compression of vertices problem.

3.7 Computational complexity

The process of the algorithm can be divided into many steps. These steps, in computational order, are finding the offset regions, performing mesh smoothing, completing coarse mesh, computing Laplacian coordinate, computing mesh parameterization, transferring relief pattern and computing inverse Laplacian transform. Let n be the number of vertices of the hole region and the offset region.

Meshes used in this research are represented using Half-edge data structure. This structure can perform vertex neighborhood query in $O(1)$.

This research uses Dijkstra's algorithm to compute the offset regions. The part has linear time complexity $O(n)$.

Mesh smoothing is performed using Finite differences method. The method is an iterative process. The number of iteration can be set by the user. From the experiment, 10 – 15 iteration is enough to smooth out the relief part of the mesh.

Each iteration has linear time complexity. Thus, mesh smoothing also has linear time complexity $O(n)$.

Coarse mesh completion can be divided into three steps, hole triangulation, mesh refinement and fairing. The inputs for the triangulation step are the set of the vertex of the hole boundary. The size of this set can be approximated by $\sqrt{n}$. The triangulation step uses dynamic programming to compute the minimal area triangulation. The running time is $O(\sqrt{n}^3) = O(n^{1.5})$. Mesh refinement and fairing are iterative processes. In each iteration, the algorithm has linear time complexity.

Laplacian transform can be done in linear time complexity $O(n)$.

Mesh parameterization requires solving linear system from the least-square method. The matrix is sparse and Cholesky factorization is used in the computation. This solver has linear complexity in the number of non-zeros element of the matrix. Since, the number of non-zeros element of the matrix is $O(n)$. Mesh parameterization has linear time complexity $O(n)$.

The most time consuming part of the algorithm is in the relief transferring part. First, the k-d tree is constructed to be used for the k-nearest neighbors search. The k-d tree construction has $O(n \log n)$ complexity. The Laplacian signatures of all the vertices in the exemplar region is computed with the complexity of $O(cn \log n)$, where c is the number of sampling point. From the experiments, the regions size of 11x11 is usually enough to analyze the relief pattern. Thus, c is constant. The $\log n$ term is the complexity of the k-nearest search. In transferring relief, each vertex of is $R_{L.}.hole$ compared with the vertex of $R_{L.}.offset$. The complexity for all comparisons is $O(cn^2)$. The complexity of the relief transferring part is $O(n^2)$.

Inverse Laplacian transform requires solving linear system from the least-square method. The matrix of this linear system is different from that used in mesh parameterization but it is also sparse. Cholesky factorization is also used as the solver. The complexity is $O(n)$.

Thus, the whole process has quadratic complexity.

CHAPTER IV

EXPERIMENTAL RESULTS

4.1 Implementation

The algorithm is implemented using Visual C++ and DirectX 9.0. This research uses Half-edge data structure from OpenMesh library [63] for mesh representations. This research also uses linear system solver from TAUCS library [32] when computing mesh parameterization and inverse Laplacian transformation. Cholesky factorization method is used for the computations since it is faster and require lower memory than other methods such as Iterative solver or Multigrid solver [49]. The Cholesky factorization method has linear complexity in the number of non-zeros element of the matrix. Since the matrices are sparse, the Compressed sparse column (CSC) is used for matrix representations [64]. CSC representation can reduce the memory consumption from $O(n^2)$ to $O(n)$ where n is the number of vertex. In Laplacian signature computation, this research uses ANN library [65] for the k-nearest neighborhood search. Euclidean distance is used to determine the closeness of the neighborhood. This research uses geometric deviation and normal deviation from PolyMeCo software [66] for mesh comparison.

4.2 Experiment and Results

The experimental setup is consisted of three parts. In the first part, the algorithm is tested with planar surfaces inputs. On this part, the inputs are test with variation in parameters of Laplacian signature distance matrices, neighborhoods sampling size and the number of nearest neighbor. The results are shown with their corresponding Laplacian signatures. The planar surfaces are used because it is easier to verify the quality of the transferred pattern of planar surfaces than that of arbitrary surfaces. The goal of this part is to determine how well the relief pattern are transferred from the offset region by these parametric setting.

In the second part, the algorithm is tested with curve surface inputs. In this part, the coarse meshes, the relief meshes and the filled meshes are presented side by side. The goal of this part is to test the features of coarse meshes computation

and the extraction of relief meshes. It is also designed to determine how result meshes depend on the quality of coarse meshes computation.

In the third part, the algorithm is tested with the scanned surfaces inputs. The inputs are scanned from real-world objects as oppose to the first part and the second part where the inputs are generated using model authoring tools. The goal of this part is to determine how well the algorithm can handle these real-world data.

In the first part of the experiments, surfaces in Figure 4.1 – 4.4 are tested with different Laplacian signature distance metrics. $Dist_{\vec{n}}$ and $Dist_H$ can be viewed as $Dist_{H\vec{n}}$ with the weight setting to one and zero respectively as in the equation.

$$Dist_{H\vec{n}} = w * Dist_{\vec{n}} + (1-w) * Dist_H$$

From Figure 4.1 – 4.4, the images (b) use distance metric $Dist_{\vec{n}}$ or $Dist_{H\vec{n}}$ with the weight of value one. The images (d) use distance metric $Dist_H$ or $Dist_{H\vec{n}}$ with the weight setting to zero. The images (f) use distance metric $Dist_{H\vec{n}}$ with the weight setting to 0.5.

From Figure 4.1 – 4.4, it can be concluded that the quality of the results does not depend on the distance metrics of the Laplacian signature. The visualizations of the Laplacian signature review that normal vector component and mean curvature component have some correspondence with each other. If the two points have the similar value of normal vector component, they are also likely to have the similar mean curvature component value. All patterns in the experiment, near-regular pattern (Figure 4.1 – 4.2), irregular pattern (Figure 4.3) and stochastic pattern (Figure 4.4) exhibit these type of relationship. For irregular pattern of the input surface in Figure 4.3, the relief patterns on the hole regions are slightly different from each others. These differences come from the irregularity nature of the surfaces.

Surfaces in Figure 4.5 – 4.8 are tested with different neighborhood size. The images (a) in Figure 4.5 – 4.8 show the input meshes with known relief patterns. The relief patterns are visualize as height maps using gray scale images. For near-regular surface of Figure 4.5, the small neighborhood size of 7x7 (30% of the relief pattern size) and 11x11 (70% of the relief pattern size) fails to capture the relief pattern

of the inputs surface. This surface requires the neighborhood size of at least 13x13 (100% of the relief pattern size) in order to faithfully capture the relief detail. The near-regular surface in figure 4.6 also has some difficulties with the neighborhood size of 7x7 and 11x11. The reconstructed relief patterns contain some bumps. In contrast with the surfaces in figure 4.7 – 4.8, irregular and stochastic patterns can be detected and reconstructed with small neighborhood size of 7x7. The bigger neighborhood sizes of 11x11 or 13x13 do not increase visual appearance quality of the results.

Surfaces in Figure 4.9 – 4.11 are tested with different k-nearest neighbor points. The numbers of points, k, used as nearest neighbor are three, six and nine. The bigger k value means that more points are used to average the Laplacian coordinate of a sampling point. This process is equivalent to smoothing the relief pattern. From the figure 4.9 – 4.11, it can be concluded that the different of k value do not have an effect on the quality of the results.

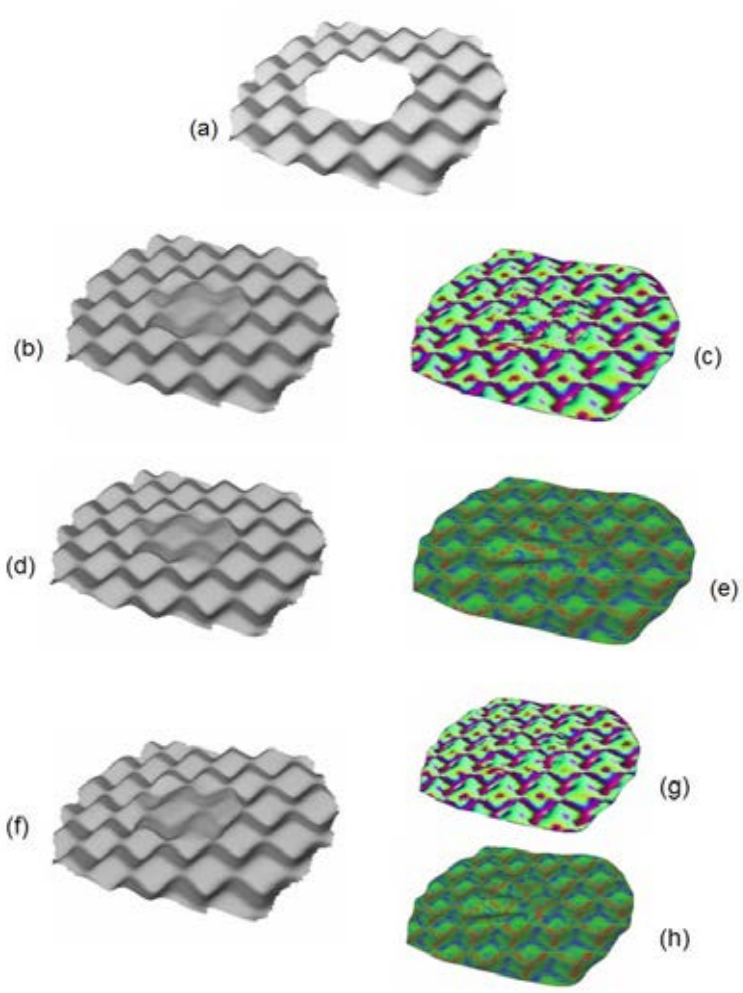

Figure 4.1: Surface completions using different distance metric in Laplacian signature comparison.

In Figure 4.1, (a) is the input mesh with near-regular relief pattern. The distance metric of the result surface (b), (d) and (f) are $Dist_{\vec{n}}$, $Dist_{H}$ and $Dist_{H\vec{n}}$ respectively. The relief completion using only normal vector components is shown in (c). The relief completion using only mean curvature components is shown in (e). The relief completion using both components is shown in (g-h).

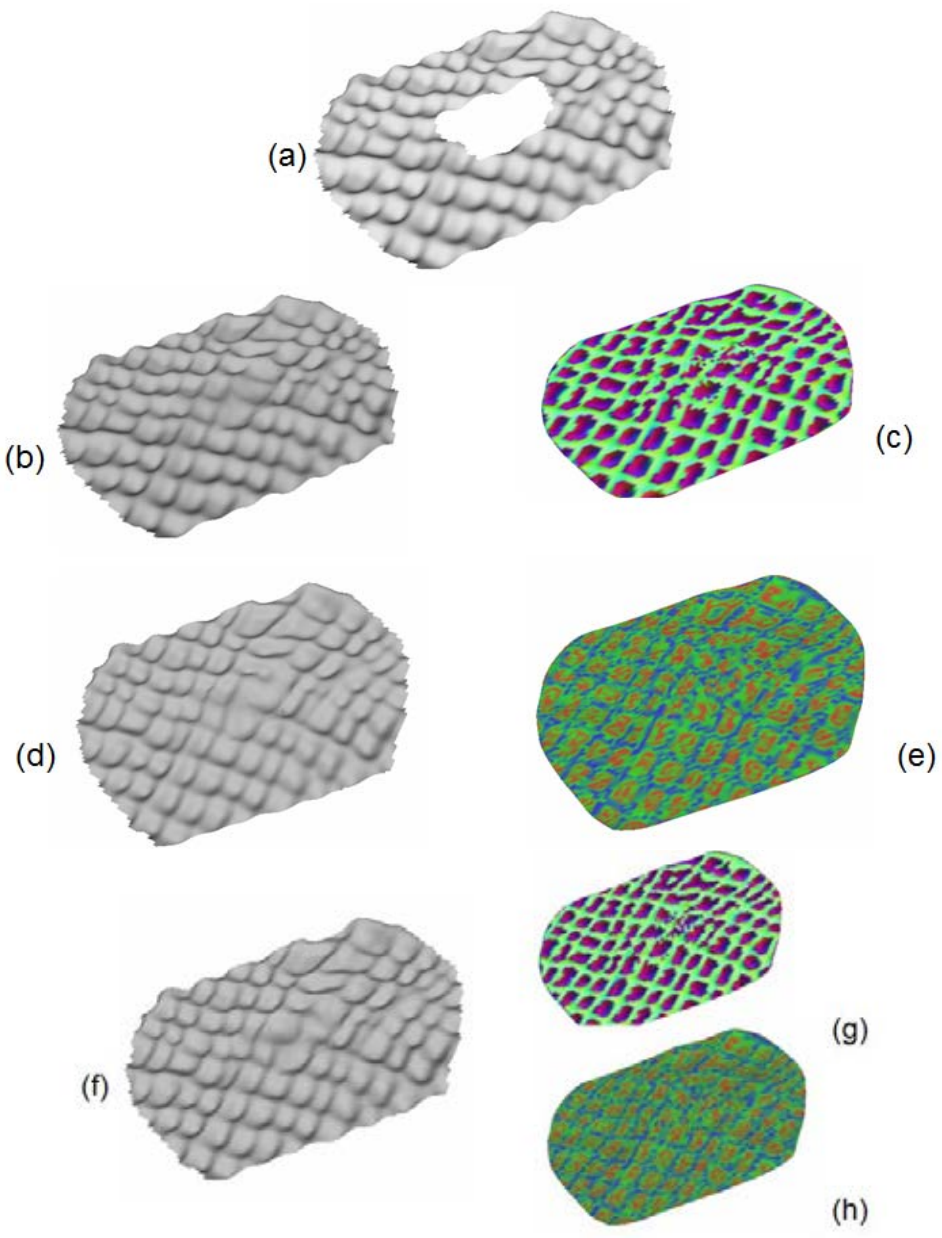

Figure 4.2: Surface completions using different distance metric in Laplacian signature comparison.

In Figure 4.2, (a) is t input mesh with near-regular relief pattern. The distance metric of the result surface (b), (d) and (f) are $Dist_{\vec{n}}$, $Dist_{H}$ and $Dist_{H\vec{n}}$ respectively. The relief completion using only normal vector components is shown in (c). The relief completion using only mean curvature components is shown in (e). The relief completion using both components is shown in (g-h).

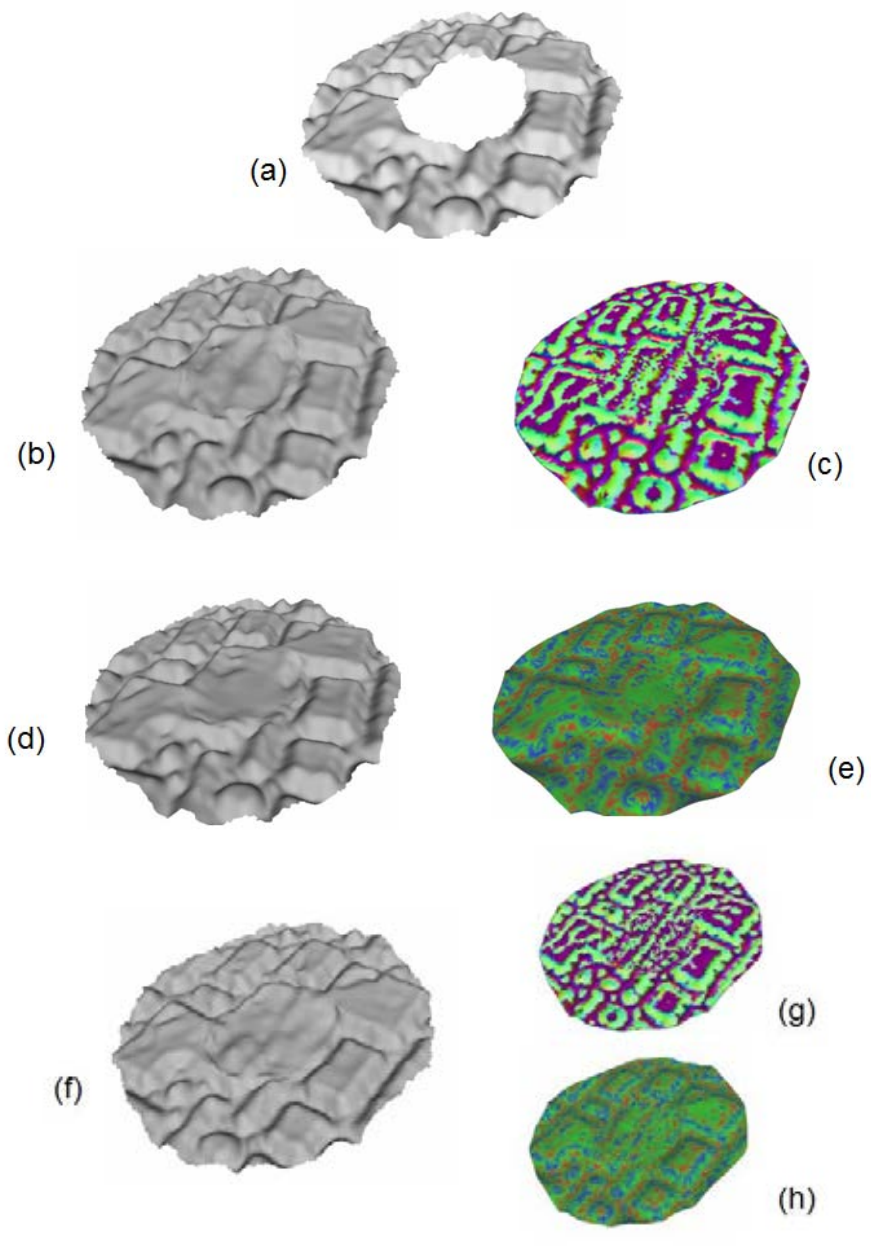

Figure 4.3: Surface completions using different distance metric in Laplacian signature comparison.

In Figure 4.3, (a) is the input mesh with irregular relief pattern. The distance metric of the result surface (b), (d) and (f) are $Dist_{\vec{n}}$, $Dist_{H}$ and $Dist_{H\vec{n}}$ respectively. The relief completion using only normal vector components is shown in (c). The relief completion using only mean curvature components is shown in (e). The relief completion using both components is shown in (g-h).

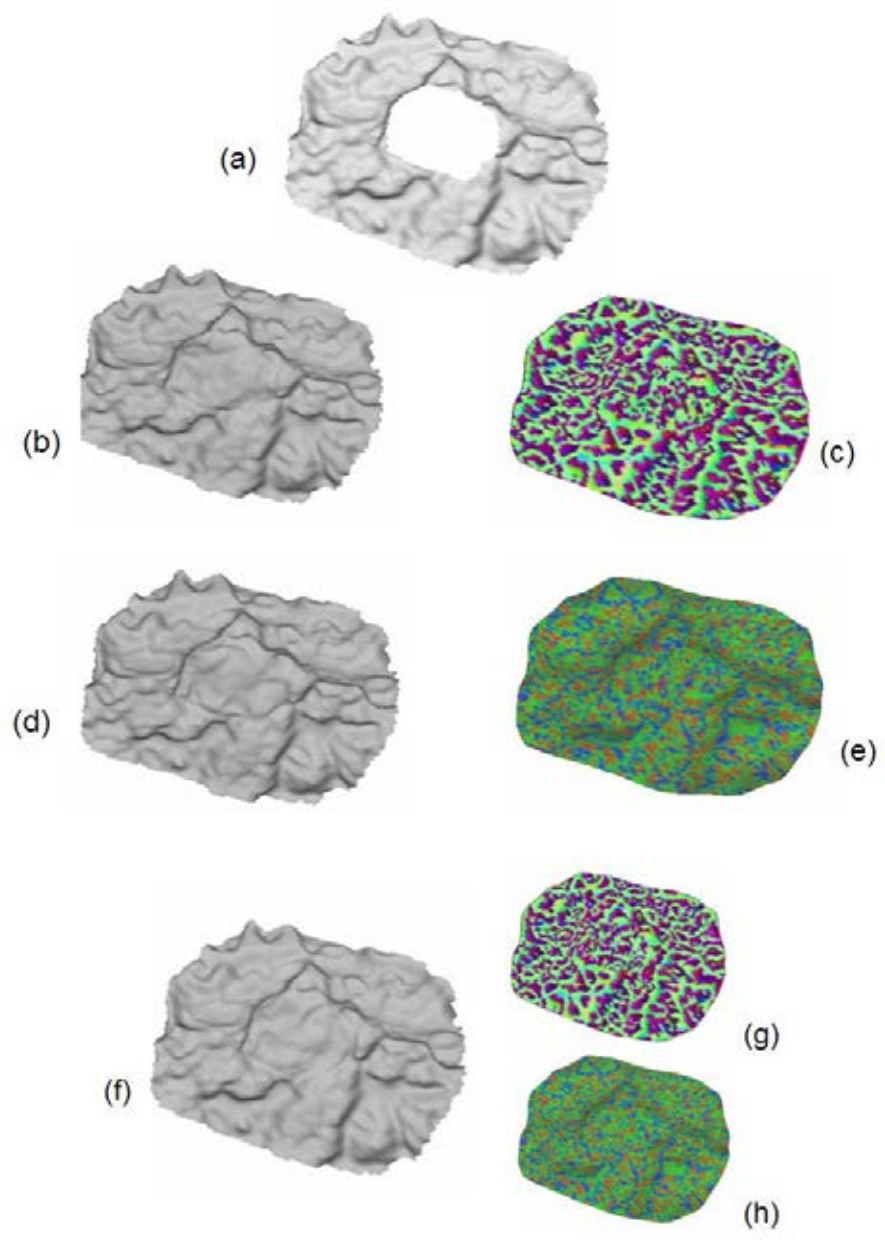

Figure 4.4: Surface completions using different distance metric in Laplacian signature comparison.

In Figure 4.4, (a) is the input mesh with stochastic relief pattern. The distance metric of the result surface (b), (d) and (f) are $Dist_{\vec{n}}$, $Dist_{H}$ and $Dist_{H\vec{n}}$ respectively. The relief completion using only normal vector components is shown in (c). The relief completion using only mean curvature components is shown in (e). The relief completion using both components is shown in (g-h).

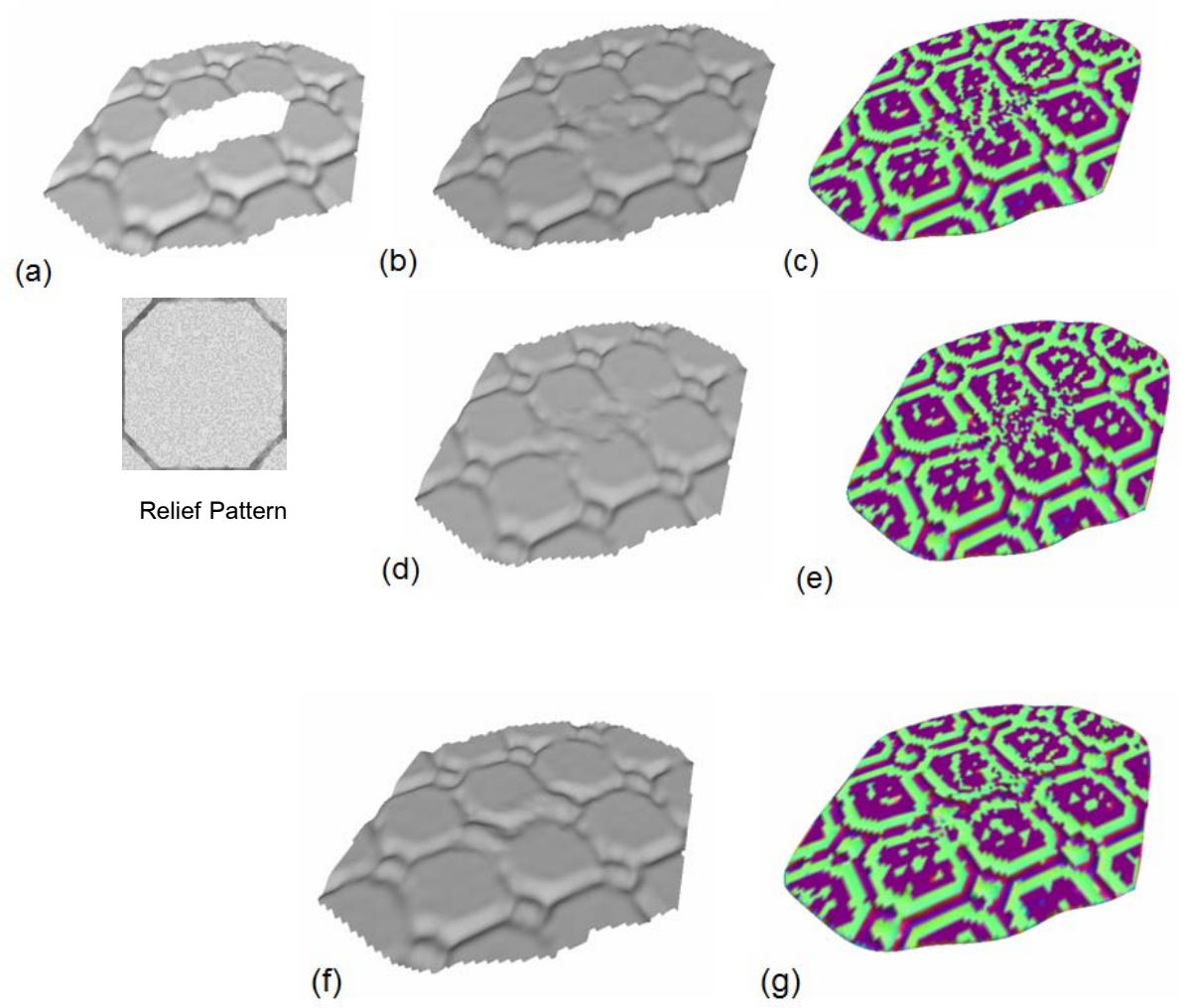

Figure 4.5: Surface completions using different neighborhood size.

In Figure 4.5, (a) is the input mesh with near-regular relief pattern. The neighborhood size of 7x7 (30% of the relief pattern size) is used for the surface completion, shown in (b), with the corresponding Laplacian signature, show in (c). The neighborhood size of 11x11 (70% of the relief pattern size) is used for the surface completion, shown in (d), with the corresponding Laplacian signature, show in (e). The neighborhood size of 13x13 (100% of the relief pattern size) is used for the surface completion, shown in (f), with the corresponding Laplacian signature, show in (g).

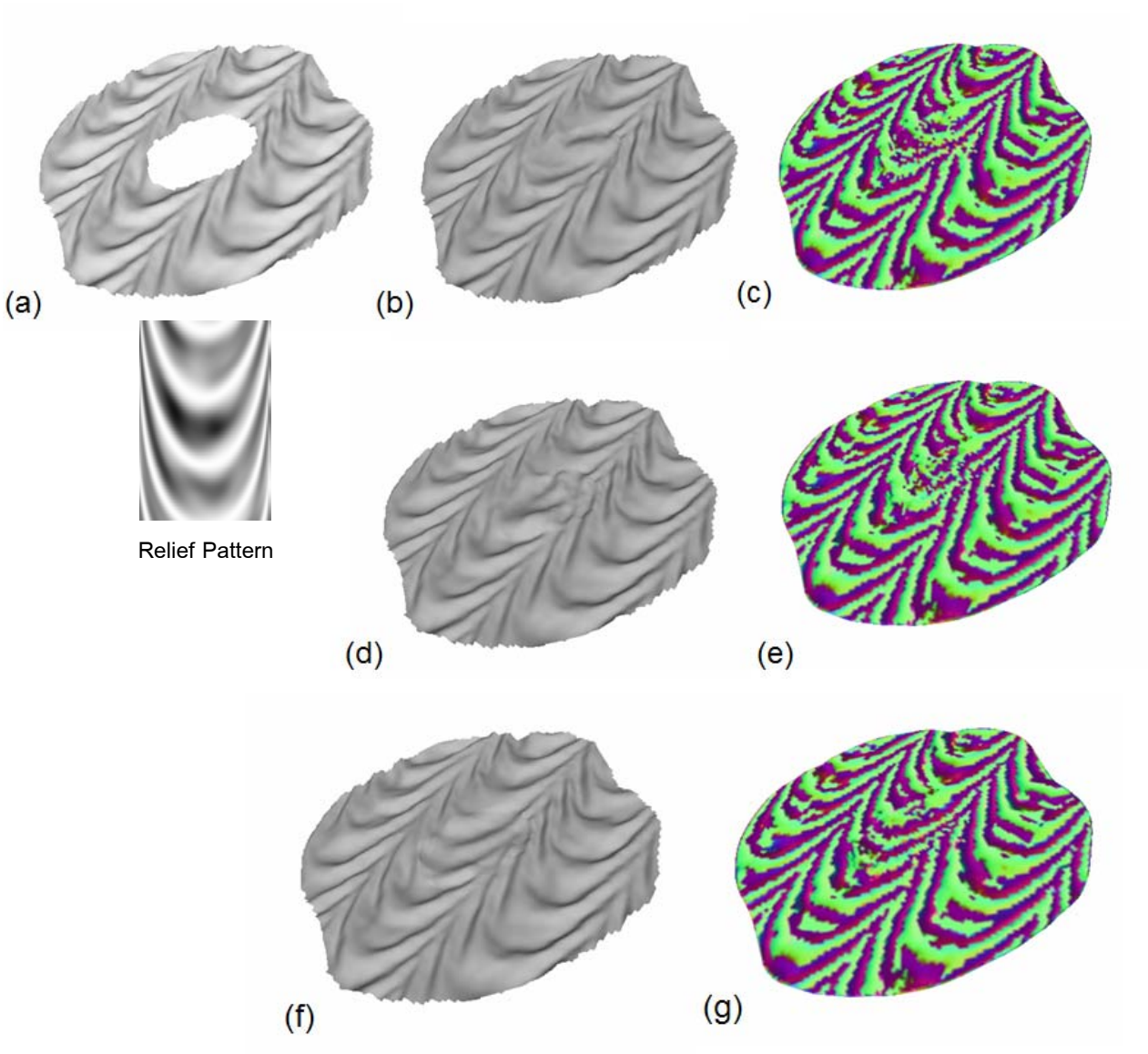

Figure 4.6: Surface completions using different neighborhood size.

In Figure 4.6, (a) is the input mesh with near-regular relief pattern. The neighborhood size of 7x7 (30% of the relief pattern size) is used for the surface completion, shown in (b), with the corresponding Laplacian signature, show in (c). The neighborhood size of 11x11 (70% of the relief pattern size) is used for the surface completion, shown in (d), with the corresponding Laplacian signature, show in (e). The neighborhood size of 13x13 (100% of the relief pattern size) is used for the surface completion, shown in (f), with the corresponding Laplacian signature, show in (g).

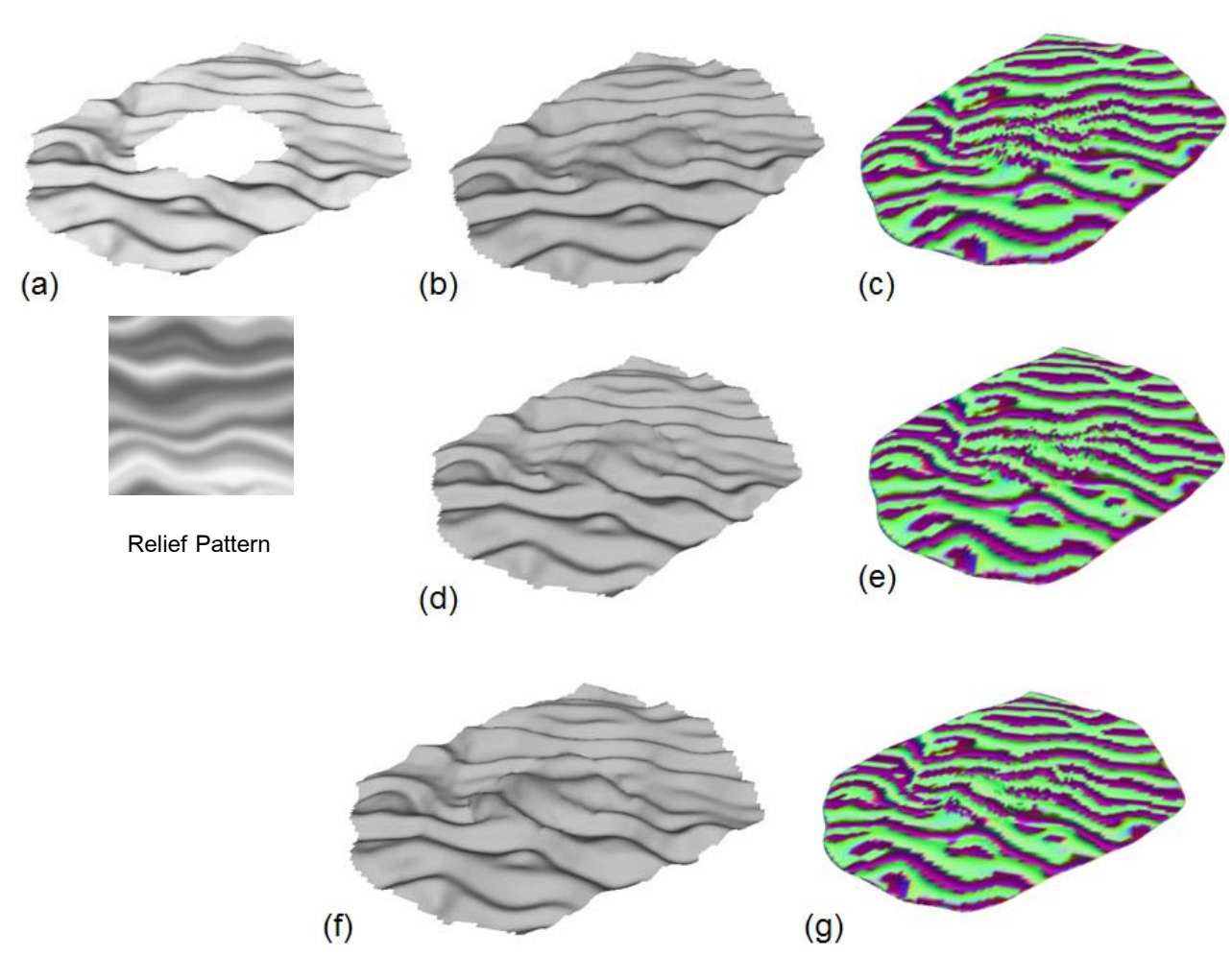

Figure 4.7: Surface completions using different neighborhood size.

In Figure 4.7, (a) is the input mesh with irregular relief pattern. The neighborhood size of 7x7 (30% of the relief pattern size) is used for the surface completion, shown in (b), with the corresponding Laplacian signature, show in (c). The neighborhood size of 11x11 (70% of the relief pattern size) is used for the surface completion, shown in (d), with the corresponding Laplacian signature, show in (e). The neighborhood size of 13x13 (100% of the relief pattern size) is used for the surface completion, shown in (f), with the corresponding Laplacian signature, show in (g).

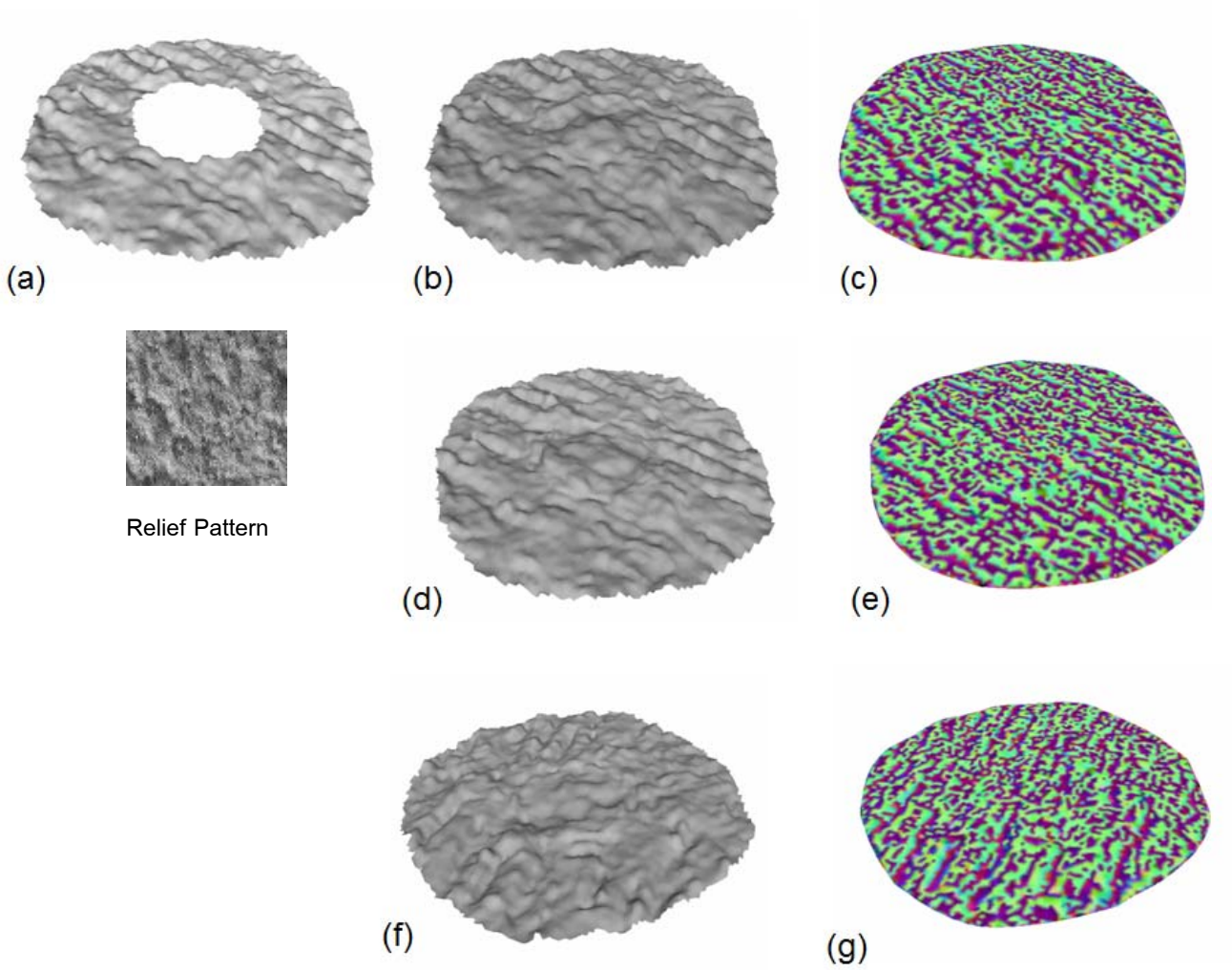

Figure 4.8: Surface completions using different neighborhood size.

In Figure 4.8, (a) is the input mesh with stochastic relief pattern. The neighborhood size of 7x7 (30% of the relief pattern size) is used for the surface completion, shown in (b), with the corresponding Laplacian signature, show in (c). The neighborhood size of 11x11 (70% of the relief pattern size) is used for the surface completion, shown in (d), with the corresponding Laplacian signature, show in (e). The neighborhood size of 13x13 (100% of the relief pattern size) is used for the surface completion, shown in (f), with the corresponding Laplacian signature, show in (g).

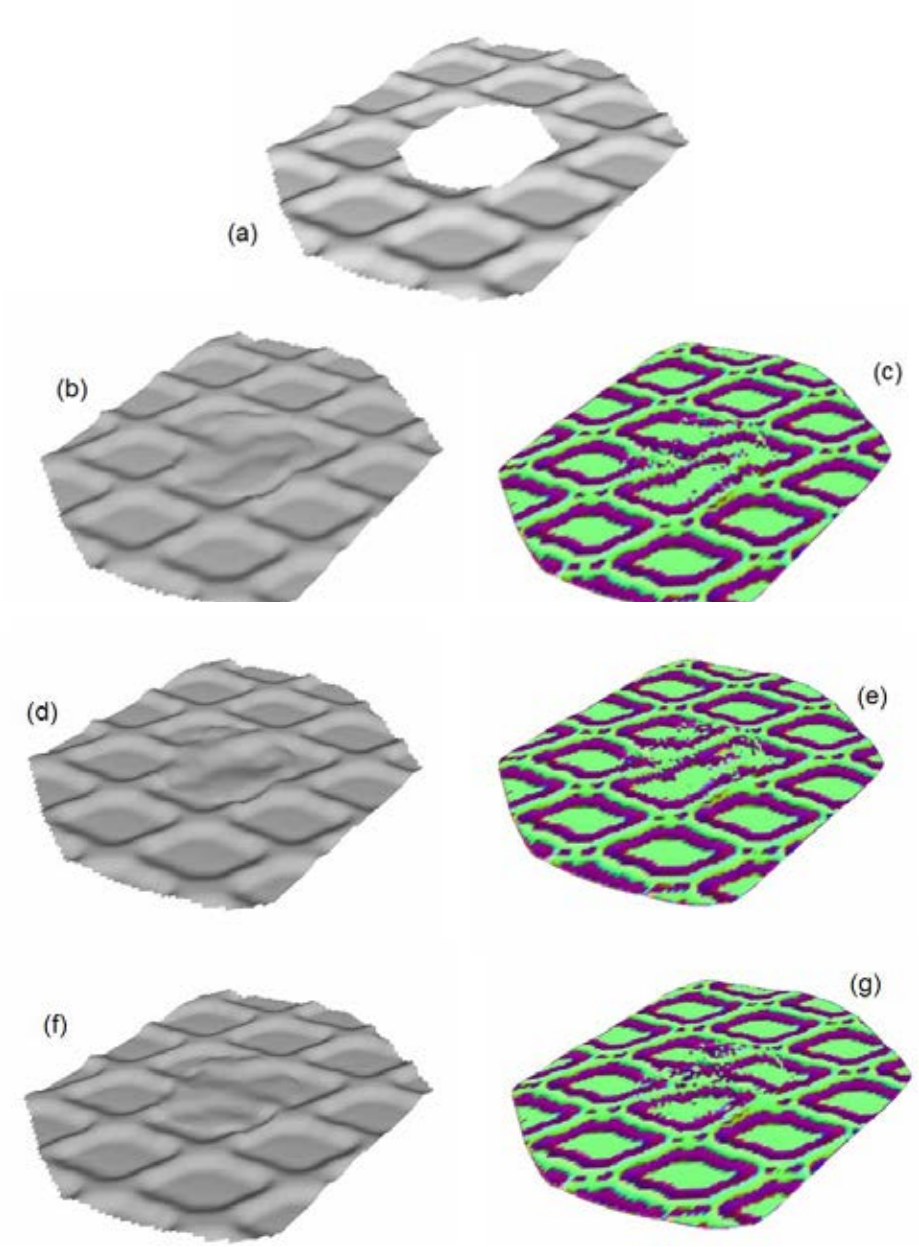

Figure 4.9: Surface completions using different k-nearest neighbor points.

In Figure 4.9, (a) is the input mesh with near-regular relief pattern. The three nearest neighbor points is used for the surface completion, shown in (b), with the corresponding Laplacian signature, show in (c). The six nearest neighbor points is used for the surface completion, shown in (d), with the corresponding Laplacian signature, show in (e). The nine nearest neighbor points is used for the surface completion, shown in (f), with the corresponding Laplacian signature, show in (g).

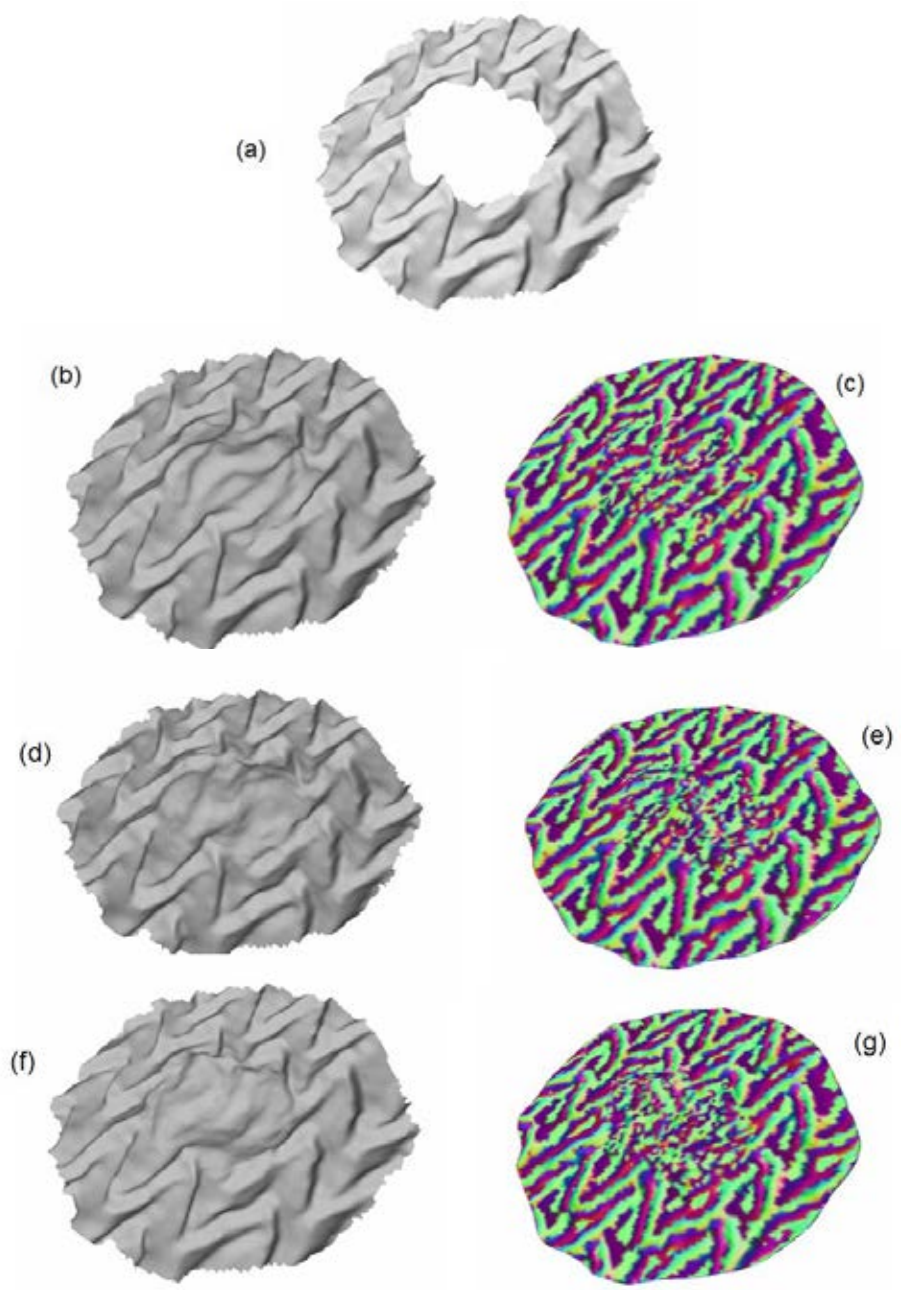

Figure 4.10: Surface completions using different k-nearest neighbor points.

In Figure 4.10, (a) is the input mesh with irregular relief pattern. The three nearest neighbor points is used for the surface completion, shown in (b), with the corresponding Laplacian signature, show in (c). The six nearest neighbor points is used for the surface completion, shown in (d), with the corresponding Laplacian signature, show in (e). The nine nearest neighbor points is used for the surface completion, shown in (f), with the corresponding Laplacian signature, show in (g).

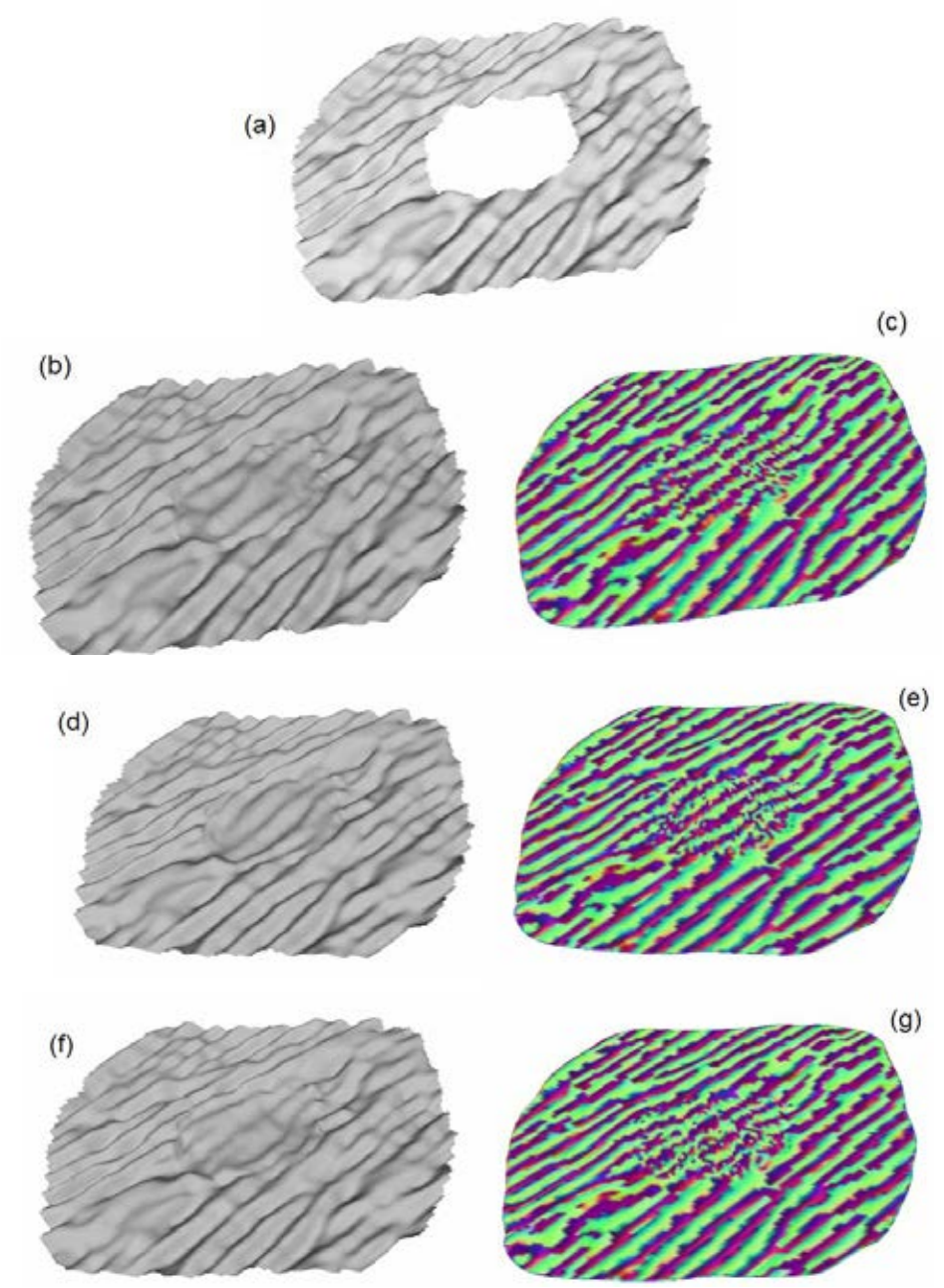

Figure 4.11: Surface completions using different k-nearest neighbor points.

In Figure 4.11, (a) is the input mesh with stochastic relief pattern. The three nearest neighbor points is used for the surface completion, shown in (b), with the corresponding Laplacian signature, show in (c). The six nearest neighbor points is used for the surface completion, shown in (d), with the corresponding Laplacian signature, show in (e). The nine nearest neighbor points is used for the surface completion, shown in (f), with the corresponding Laplacian signature, show in (g).

The second part of the experiment tests the surfaces that are non-planar. This part tests the quality of the coarse mesh and the relief patterns. In Figure 4.12, the relief patterns on the input surface (a) contain some distortion due to the curvature of the surface. The relief patterns on the curve surface are harder to extract than the relief patterns on the planar surfaces. Thus, the extracted relief pattern (c) is not as regular as the patterns in figure 4.5. In addition, there is high level of noise on the relief pattern. This is because the height of the relief pattern is very low compare to the coarse surface (b). The reconstructed surface (d) does not capture the relief pattern of the input surface.

In figure 4.13, the relief pattern (c) contains low level of noise. The relief pattern clearly stands out from the coarse mesh. Thus, the algorithm can successfully filled the relief information to the hole region and can successfully reconstruct the surface (d).

In figure 4.14, the input surface (a) is very rough. The relief pattern is stochastic. The coarse mesh (b) is computed using 15 iteration of mesh smoothness. The result coarse mesh still looks bumpy. However, the reconstructed surface (d) looks promising because of the stochastic nature of the pattern.

The surface in figure 4.15 contains convex and concave curvatures. The algorithm can produce satisfying result coarse mesh (b). The relief pattern is clearly visible and contains low level of noise (c). Thus, the surface can be filled with relief pattern successfully (d).

It can be conclude that the relief patterns of curve surfaces must be clearly distinct from the underlying coarse mesh in order for the relief extraction to be successful. The curvature of the surface can distort the regularity of the patterns making it hard to detect the patterns.

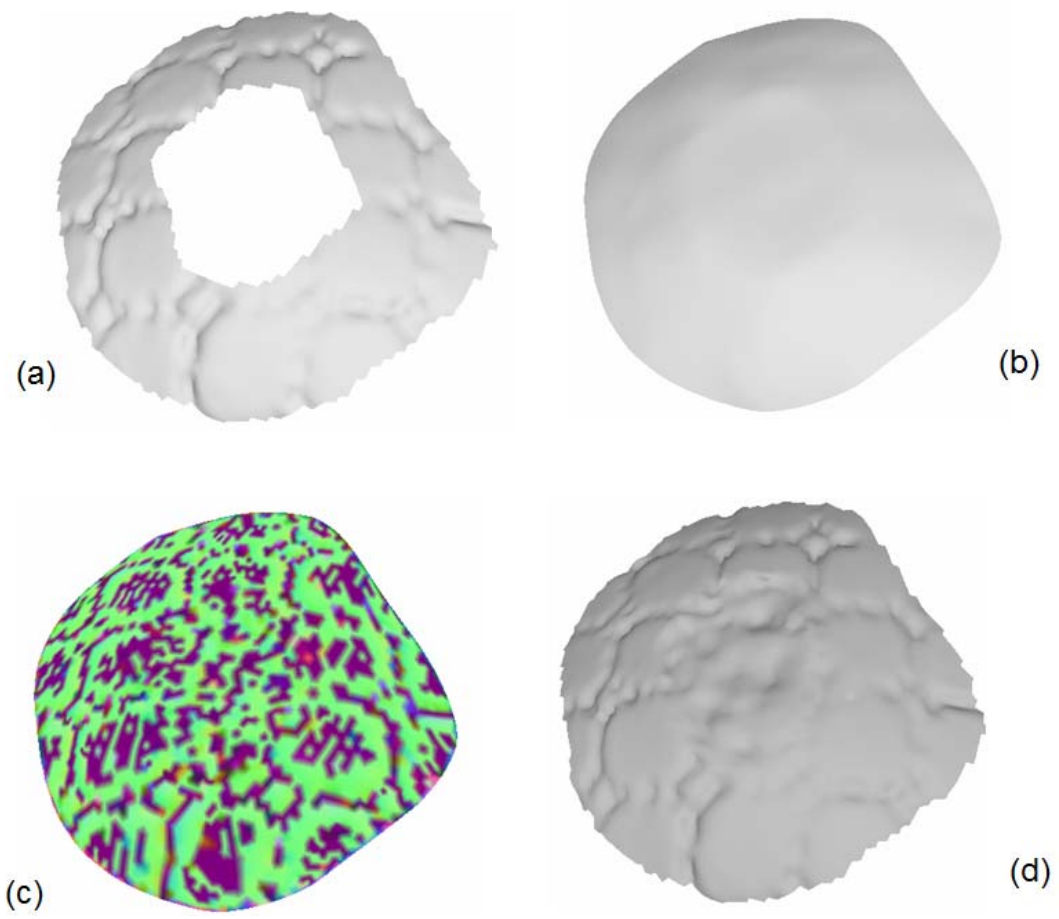

Figure 4.12: Surface completions on a curve surface.

In Figure 4.12, (a) is the input mesh with near-regular relief pattern. (b) is the coarse mesh completion. (c) is the relief mesh represented in Laplacian coordinate. The relief pattern is extracted from the offset region and is transferred to the hole region. (d) is the reconstructed surface using Inverse Laplacian transformation. The neighborhood size of 15x15 is used in the computation.

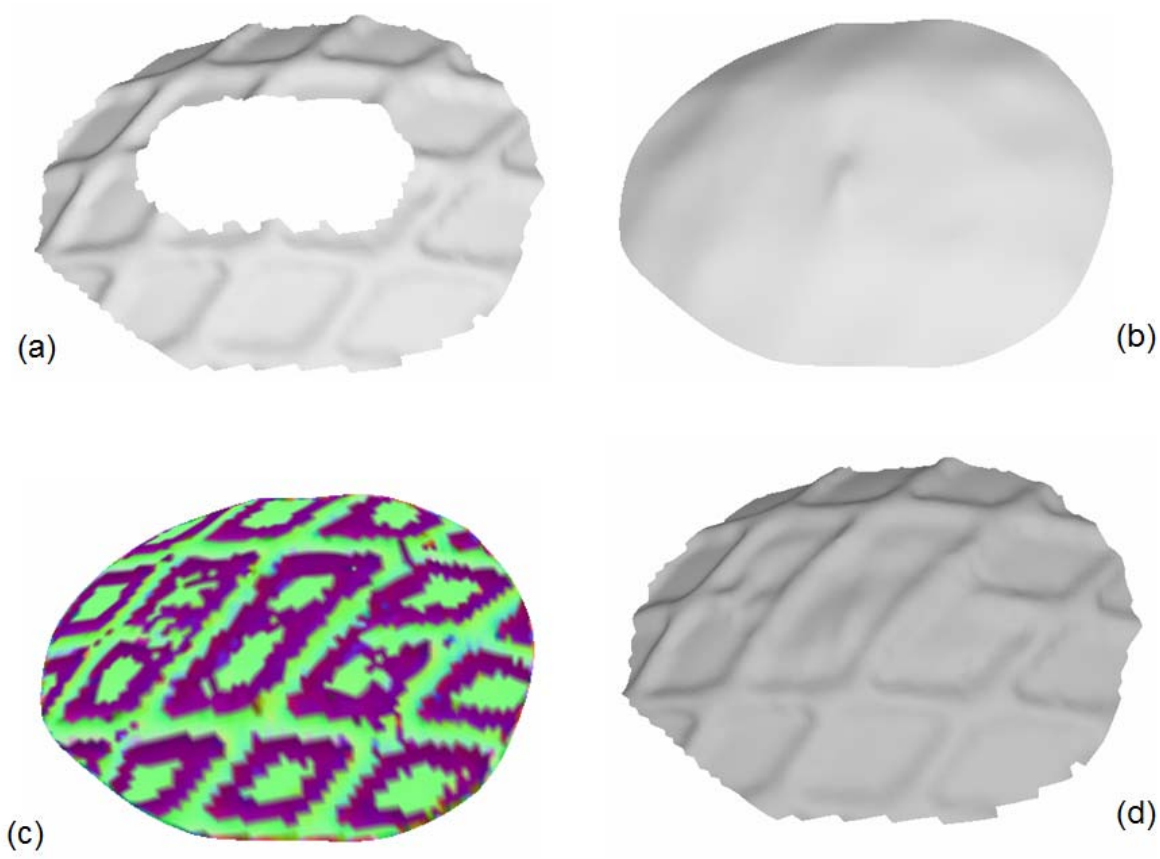

Figure 4.13: Surface completions on a curve surface.

In Figure 4.13, (a) is the input mesh with near-regular relief pattern. (b) is the coarse mesh completion. (c) is the relief mesh represented in Laplacian coordinate. The relief pattern is extracted from the offset region and is transferred to the hole region. (d) is the reconstructed surface using Inverse Laplacian transformation. The neighborhood size of 15x15 is used in the computation.

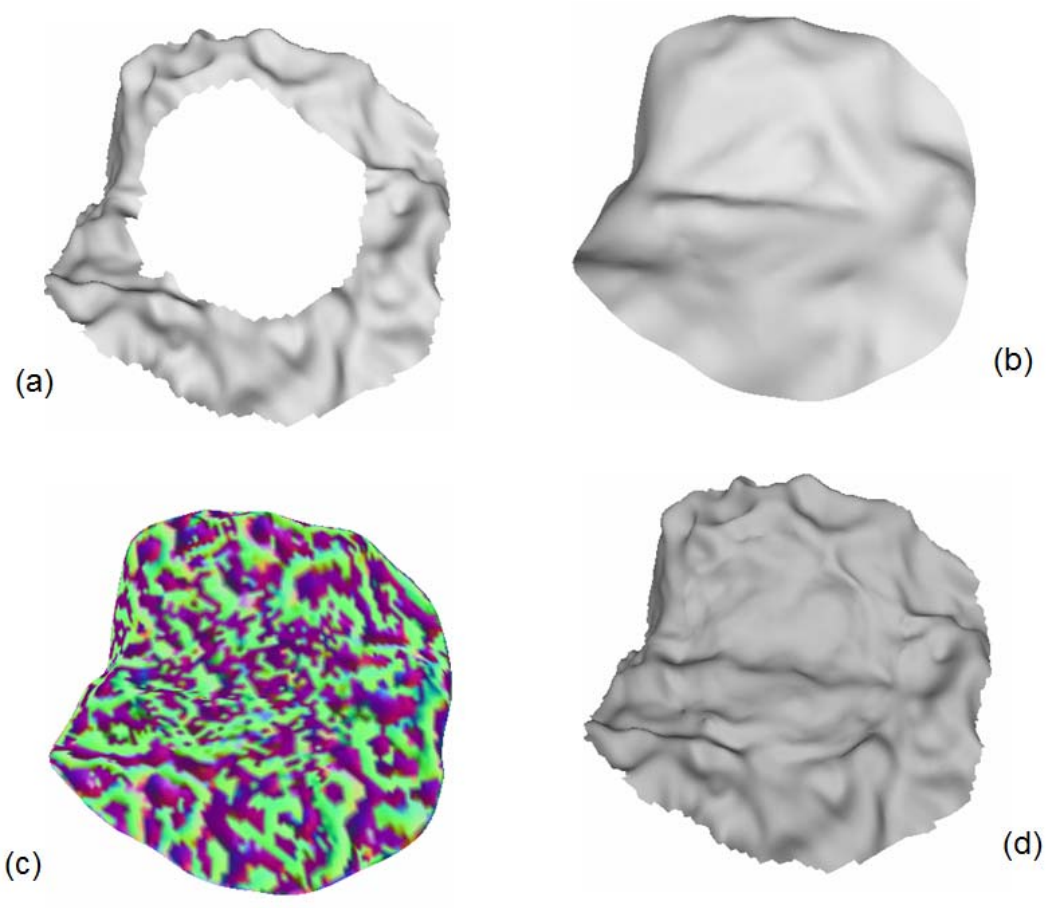

Figure 4.14: Surface completions on a curve surface.

In Figure 4.14, (a) is the input mesh with stochastic relief pattern. (b) is the coarse mesh completion. (c) is the relief mesh represented in Laplacian coordinate. The relief pattern is extracted from the offset region and is transferred to the hole region. (d) is the reconstructed surface using Inverse Laplacian transformation. The neighborhood size of 11x11 is used in the computation.

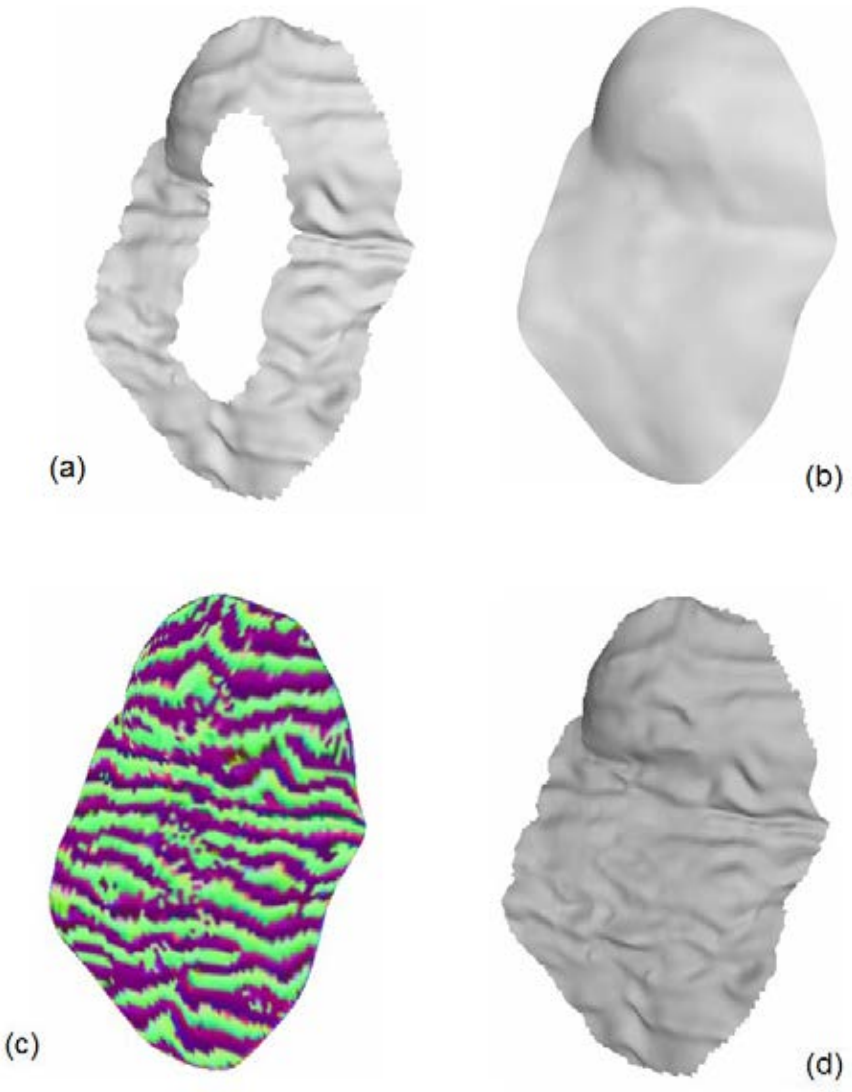

Figure 4.15: Surface completions on a curve surface.

In Figure 4.15, (a) is the input mesh with irregular relief pattern. (b) is the coarse mesh completion. (c) is the relief mesh represented in Laplacian coordinate. The relief pattern is extracted from the offset region and is transferred to the hole region. (d) is the reconstructed surface using Inverse Laplacian transformation. The neighborhood size of 11x11 is used in the computation.

In the third part, the algorithm is tested with the scanned surfaces inputs. Only the normal vector component of the signature is used for the surface comparison. This should be enough since the mean curvature component follow the normal vector component. The k-nearest neighbor of three is used in the experiments since the higher the number of k value introduces more computational time but not increase the quality of the results.

The Stanford bunny model in figure 4.16 – 4.17 contains stochastic relief pattern. Thus, the neighborhood size of 7x7 should be enough to capture the pattern characteristic. In figure 4.16, the final mesh (h) looks significantly different from the reference mesh (f). This is because the coarse mesh (g) cannot produce the leg structure of the bunny. The hole on the coarse mesh is filled with minimal surface area and minimal normal variation. These geometric properties are not enough to capture the context information of the model. In figure 4.17, the input surface (a) does not contain abrupt change of the geometric structure. As a result, the coarse mesh (g) has the same structure as the reference mesh (f). The final mesh (h) appears similar to the reference mesh (f).

The armadillo leg model in figure 4.18 contains near-regular relief pattern. The neighborhood size of 13x13 is used in this experiment. The relief pattern (d) contains some noise. Due to the hole's shape, the hole of the coarse mesh is flat (g) rather than convex. The relief pattern of the result mesh (h) is not clearly visible as in the reference mesh (f) because the height of the pattern is relatively low.

The Ajax model in figure 4.19 contains irregular to stochastic relief pattern. The neighborhood size of 13x13 is used in this experiment. The input model (b) is very rough and steep. Although, the coarse mesh (c) is smoothed with 15 iterations, it still looks bumpy. The relief pattern (d) does not exhibit any repeated pattern. Thus, the filled surface (h) looks different when compare to the reference surface (g). However, the filled surface (h) has the same visual appearance as the offset region.

Figure 4.20 shows a wide range of relief patterns that can be completed using the proposed method. The bunny model (c) and dragon model (d) contain many holes with multiple relief patterns. In the dragon model, the model does not contain

large exemplar area. Thus, it is a challenging for the algorithm to extract the pattern from the available surface. However, the proposed method can produce satisfying result. The variables setting on all of these holes completion in figure 4.20 are the same. The users do not have to fine tune the parameters for each type of relief patterns on each input models.

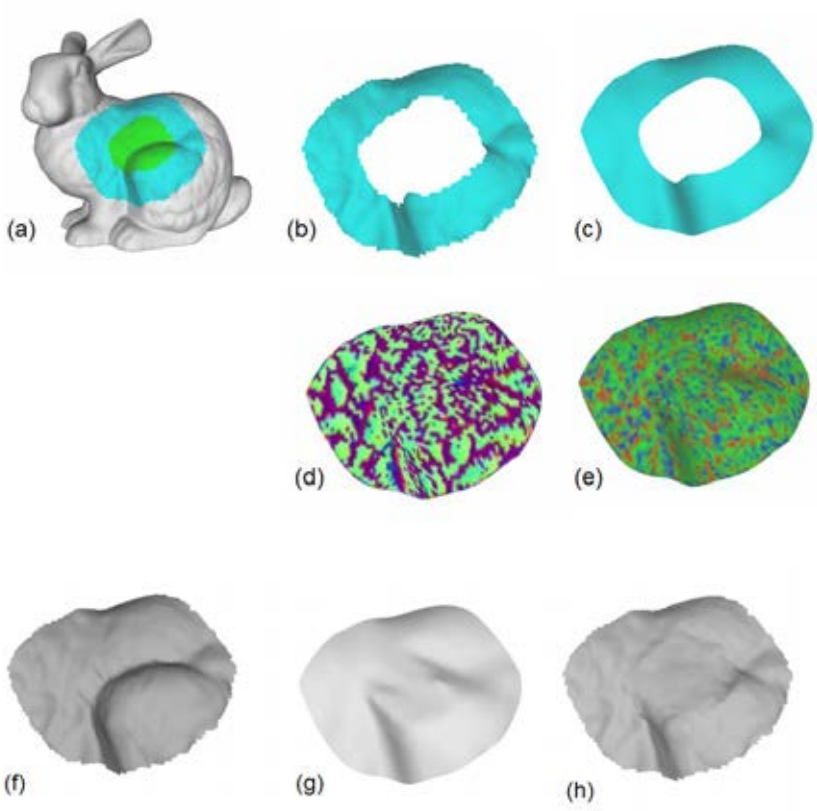

Figure 4.16: Surface completions on the Stanford bunny model.

In Figure 4.16, (a) is the input mesh with the hole region shaded in green. (b) is the offset region. (c) is the coarse mesh. (d) and (e) are the relief mesh represented in Laplacian coordinates. (g) is the smoothly filled coarse mesh. (h) is the result surface completion with relief pattern. (f) is the reference mesh extracted form the bunny model. The neighborhood size of 11x11 is used in the computation.

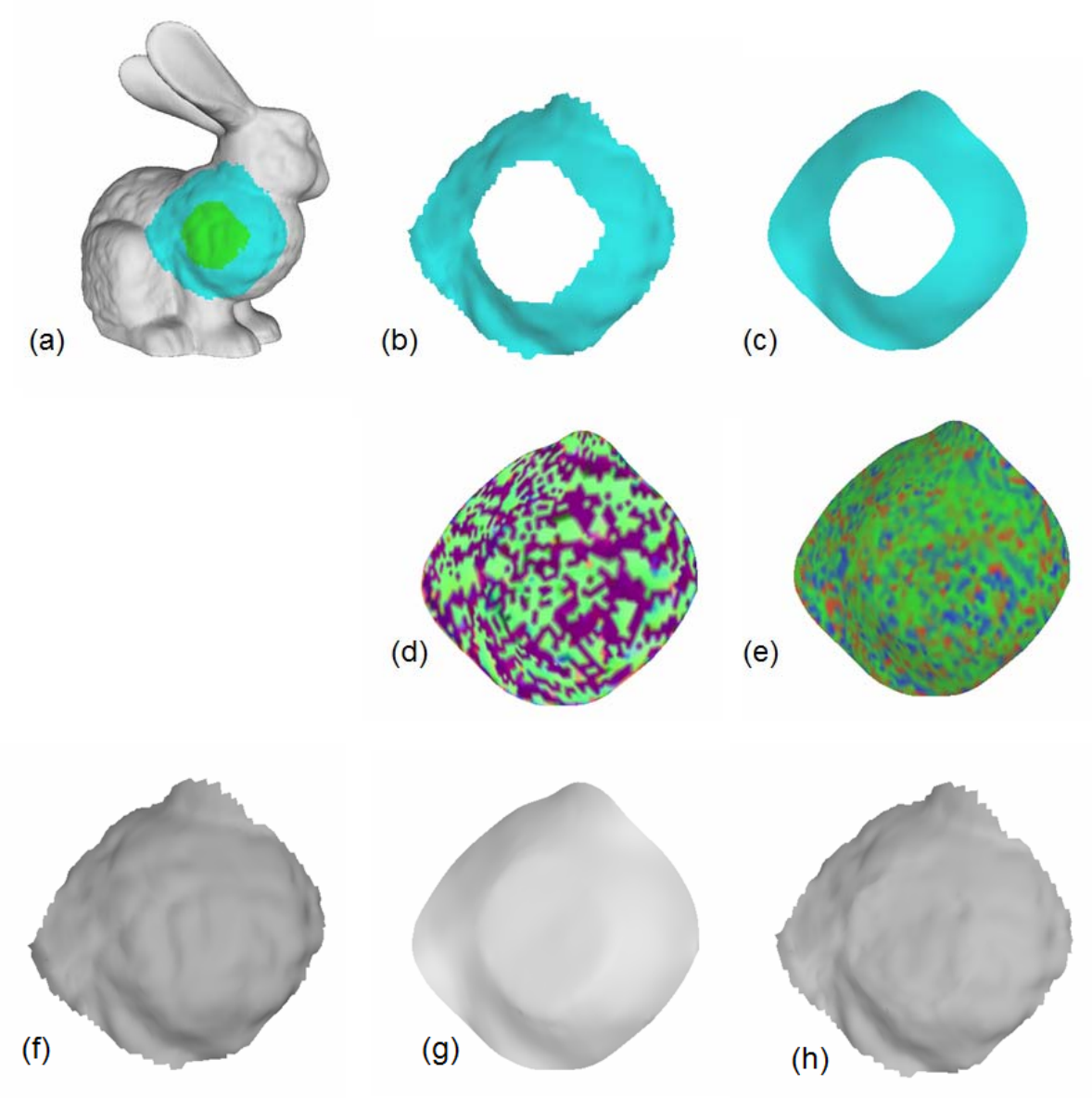

Figure 4.17: Surface completions on the Stanford bunny model.

In Figure 4.17, (a) is the input mesh with the hole region shaded in green. (b) is the offset region. (c) is the coarse mesh. (d) and (e) are the relief mesh represented in Laplacian coordinates. (g) is the smoothly filled coarse mesh. (h) is the result surface completion with relief pattern. (f) is the reference mesh extracted form the bunny model. The neighborhood size of 11x11 is used in the computation.

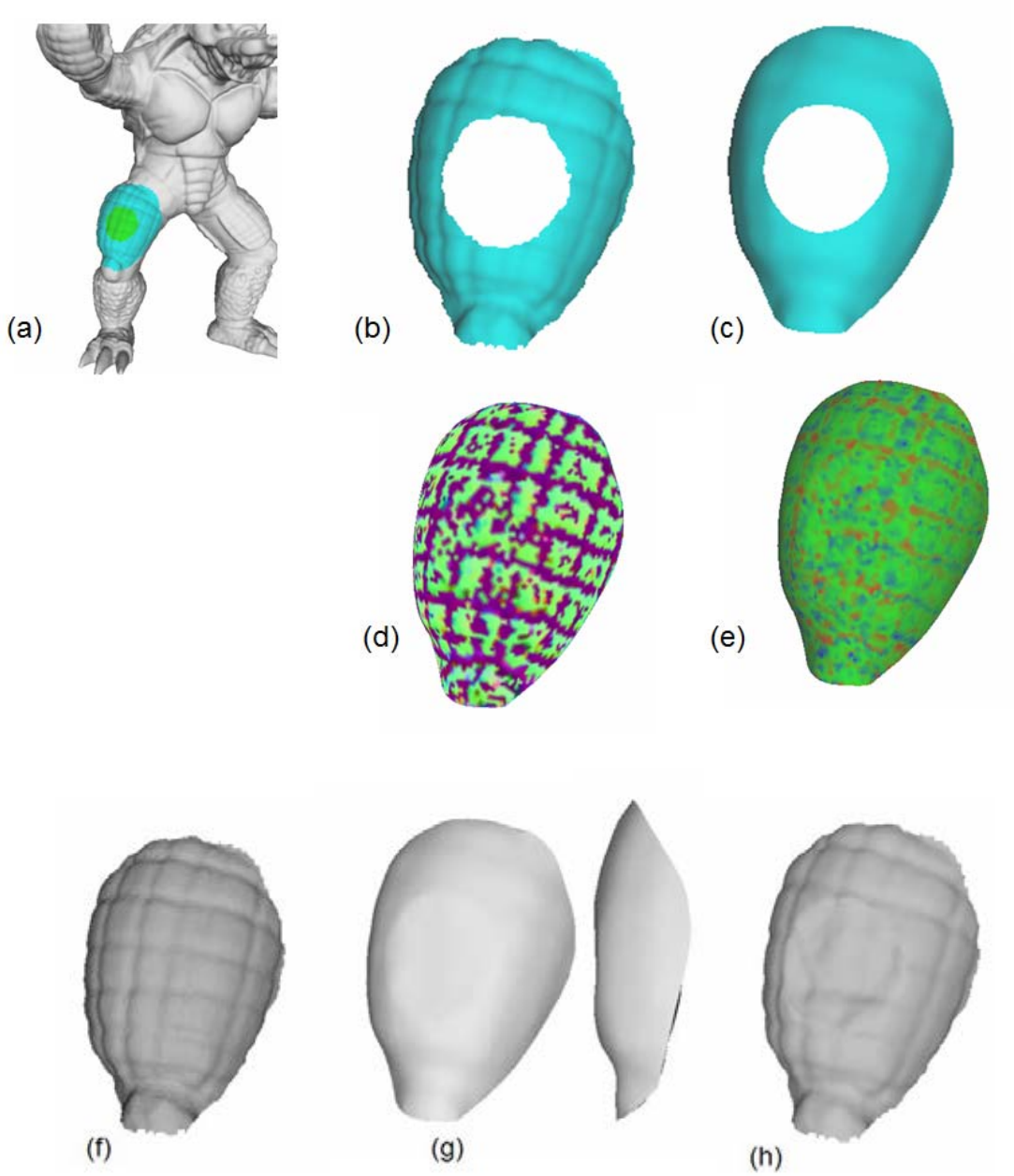

Figure 4.18: Surface completions on the Armadillo model.

In Figure 4.18, (a) is the input mesh with the hole region shaded in green. (b) is the offset region. (c) is the coarse mesh. (d) and (e) are the relief mesh represented in Laplacian coordinates. (g) is the smoothly filled coarse mesh. (h) is the result surface completion with relief pattern. (f) is the reference mesh extracted form the armadillo model. The neighborhood size of 13x13 is used in the computation.

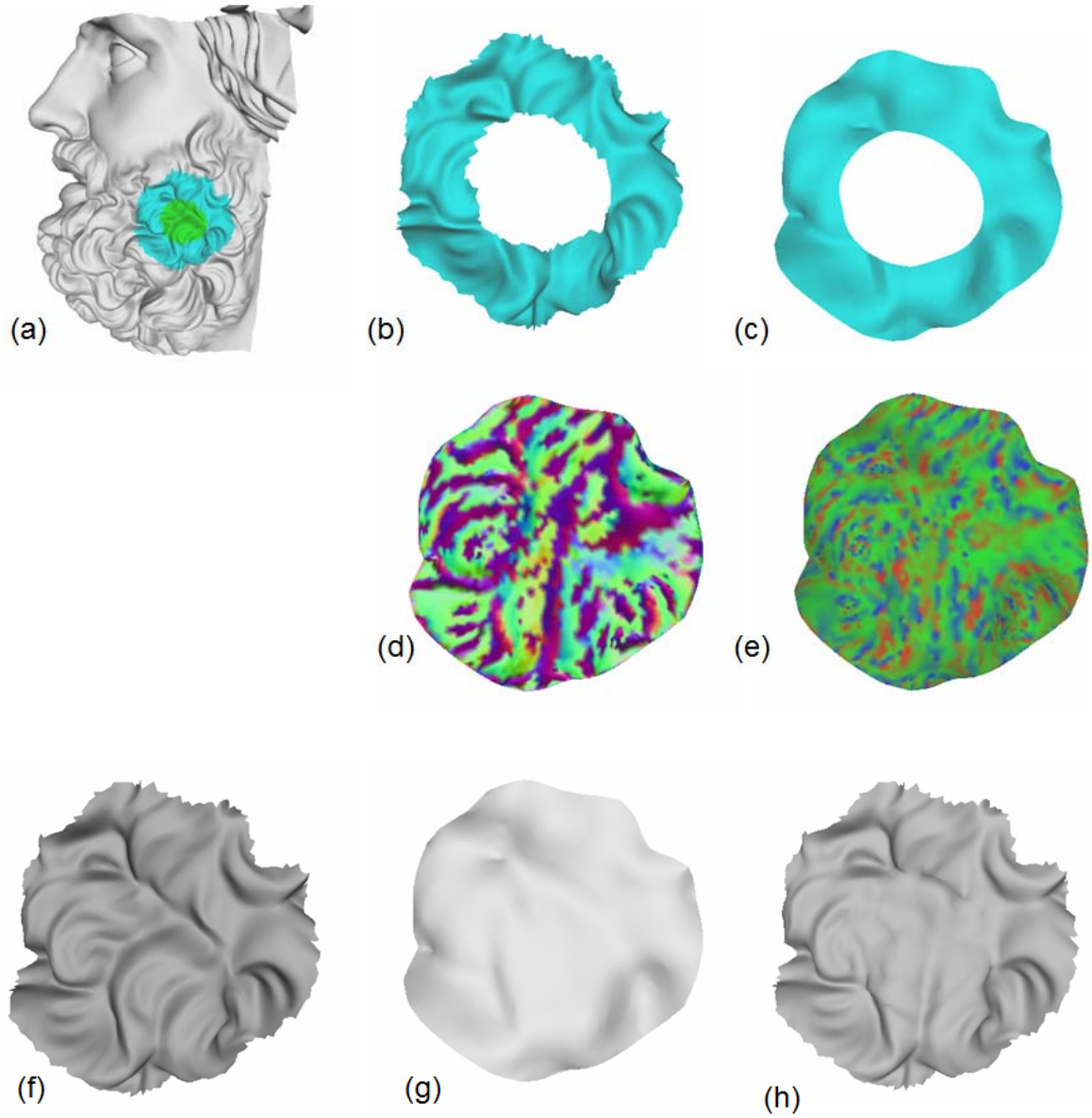

Figure 4.19: Surface completions on the Ajax model.

In Figure 4.19, (a) is the input mesh with the hole region shaded in green. (b) is the offset region. (c) is the coarse mesh. (d) and (e) are the relief mesh represented in Laplacian coordinates. (g) is the smoothly filled coarse mesh. (h) is the result surface completion with relief pattern. (f) is the reference mesh extracted form the ajax model. The neighborhood size of 13x13 is used in the computation.

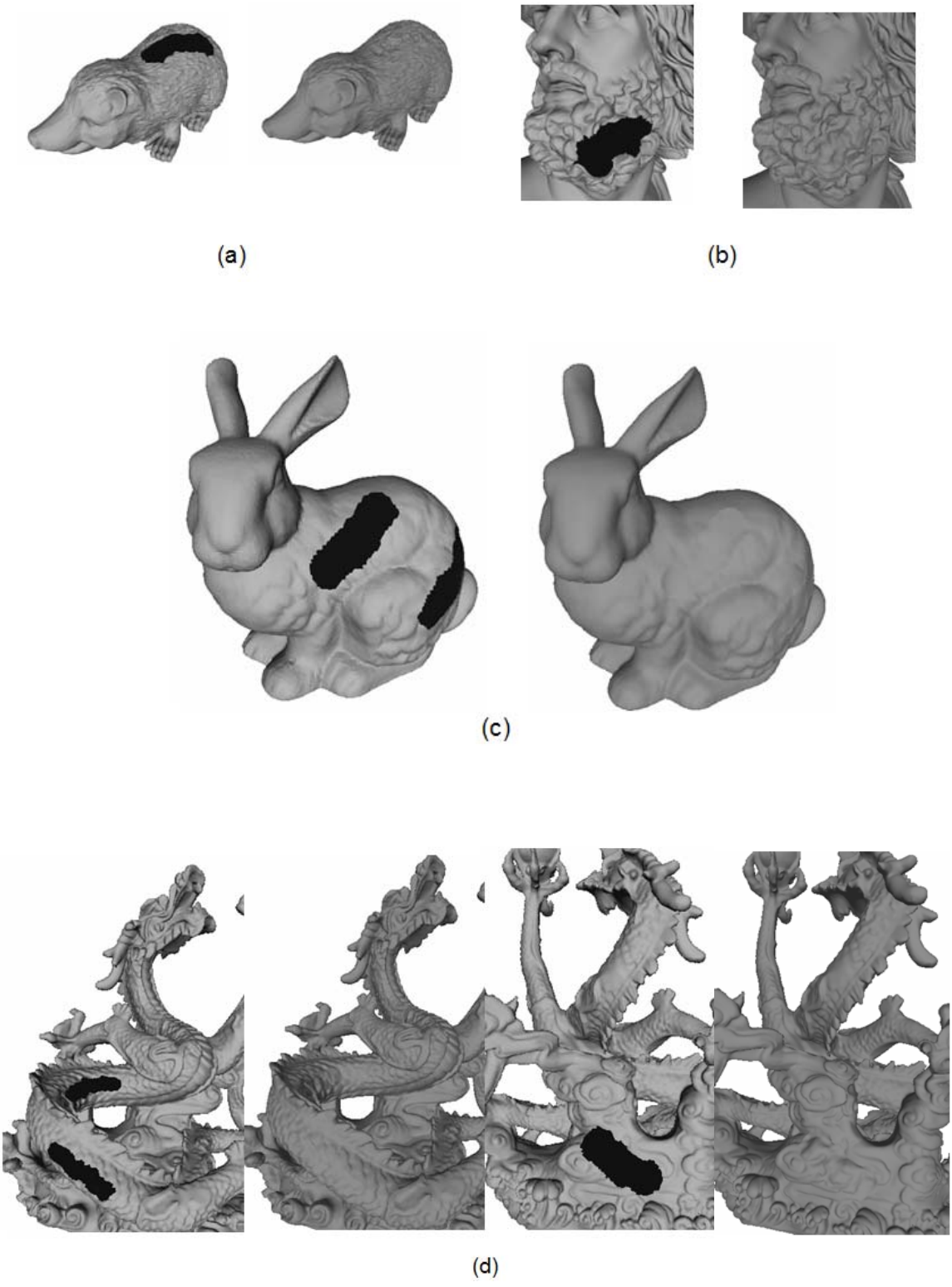

Figure 4.20: Surface completion using Laplacian transform on various models.

The goal of this research is to produce the filled surface that have similar in visual appearance with the original surface. The synthesized patterns do not need to be the exact replicate of the original patterns. In Figure 4.21, surface (a) is the original surface. Surface (b) is the surface completed with relief transfer using the proposed algorithm. Surface (e) is the smoothly filled surface. Image (c) and (d) show the Geometric deviation of 3.67 and Normal deviation of 1.60 when comparing surface (b) to surface (a). Image (f) and (g) show the Geometric deviation of 4.21 and Normal deviation of 1.33 when comparing surface (e) to surface (a). The results indicate that the errors value from these matrices do not reflect the visual appearance of the comparing surfaces. The error metrics generally used for the mesh comparison, such as Geometric deviation or Normal deviation [66], cannot be used in this research.

In addition, these metrics are transformation variant. The two surfaces with the same geometric pattern but are different in transformation, such as translation or rotation, would have great error.

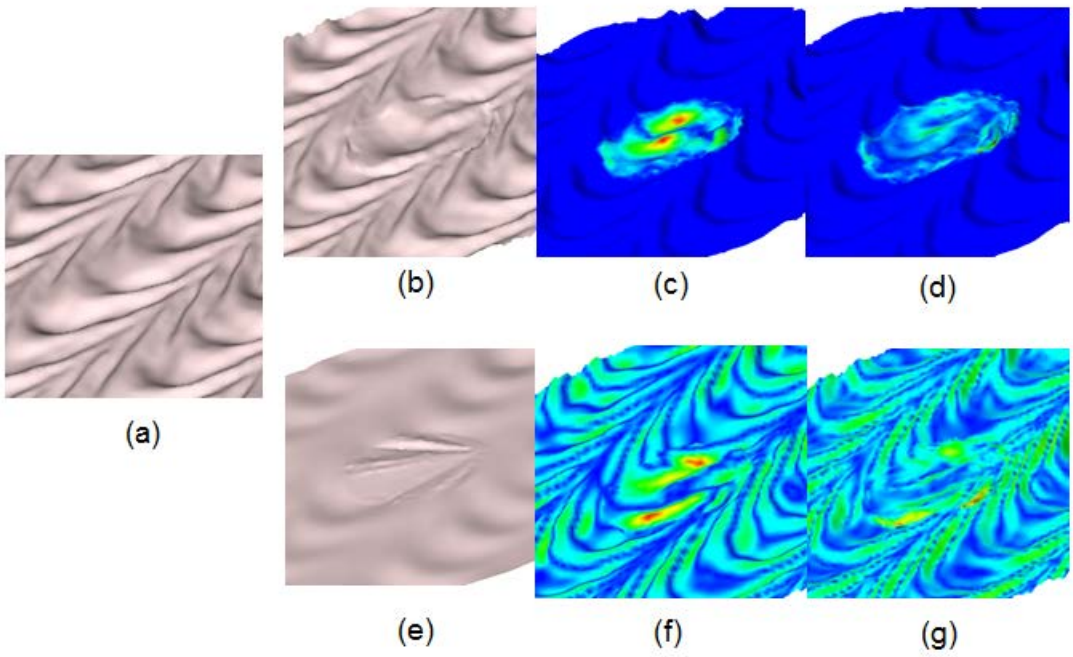

Figure 4.21: The geometric deviation and the normal deviation of input surfaces.

Table 4.1: The computation time of all the models in the experiments.

Model	No. of Vertices	Distance metric	Neighborhood size	k-nearest size	Computation Time (seconds)
1. Figure 4.1 (b)	8,075	$Dist_{\vec{n}}$	13x13	3	549
2. Figure 4.1 (d)	8,075	$Dist_{H}$	13x13	3	534
3. Figure 4.1 (f)	8,075	$Dist_{H\vec{n}}$	13x13	3	562
4. Figure 4.2 (b)	6,095	$Dist_{\vec{n}}$	13x13	3	318
5. Figure 4.2 (d)	6,095	$Dist_{H}$	13x13	3	320
6. Figure 4.2 (f)	6,095	$Dist_{H\vec{n}}$	13x13	3	323
7. Figure 4.3 (b)	9,911	$Dist_{\vec{n}}$	13x13	3	876
8. Figure 4.3 (d)	9,911	$Dist_{H}$	13x13	3	870
9. Figure 4.3 (f)	9,911	$Dist_{H\vec{n}}$	13x13	3	880
10. Figure 4.4 (b)	6,298	$Dist_{\vec{n}}$	13x13	3	356
11. Figure 4.4 (d)	6,298	$Dist_{H}$	13x13	3	349
12. Figure 4.4 (f)	6,298	$Dist_{H\vec{n}}$	13x13	3	361
13. Figure 4.5 (b)	5,860	$Dist_{\vec{n}}$	7x7	3	90
14. Figure 4.5 (d)	5,860	$Dist_{\vec{n}}$	11x11	3	150
15. Figure 4.5 (f)	5,860	$Dist_{\vec{n}}$	13x13	3	316
16. Figure 4.6 (b)	6,746	$Dist_{\vec{n}}$	7x7	3	111
17. Figure 4.6 (d)	6,746	$Dist_{\vec{n}}$	11x11	3	182
18. Figure 4.6 (f)	6,746	$Dist_{\vec{n}}$	13x13	3	379
19. Figure 4.7 (b)	7,254	$Dist_{\vec{n}}$	7x7	3	140
20. Figure 4.7 (d)	7,254	$Dist_{\vec{n}}$	11x11	3	235

Model	No. of Vertices	Distance metric	Neighborhood size	k-nearest size	Computation Time (seconds)
21. Figure 4.7 (f)	7,254	$Dist_{\vec{n}}$	13x13	3	459
22. Figure 4.8 (b)	6,904	$Dist_{\vec{n}}$	7x7	3	120
23. Figure 4.8 (d)	6,904	$Dist_{\vec{n}}$	11x11	3	201
24. Figure 4.8 (f)	6,904	$Dist_{\vec{n}}$	13x13	3	417
25. Figure 4.9 (b)	10,161	$Dist_{\vec{n}}$	11x11	3	502
26. Figure 4.9 (d)	10,161	$Dist_{\vec{n}}$	11x11	6	530
27. Figure 4.9 (f)	10,161	$Dist_{\vec{n}}$	11x11	9	541
28. Figure 4.10 (b)	7,135	$Dist_{\vec{n}}$	11x11	3	248
29. Figure 4.10 (d)	7,135	$Dist_{\vec{n}}$	11x11	6	275
30. Figure 4.10 (f)	7,135	$Dist_{\vec{n}}$	11x11	9	289
31. Figure 4.11 (b)	8,908	$Dist_{\vec{n}}$	11x11	3	409
32. Figure 4.11 (d)	8,908	$Dist_{\vec{n}}$	11x11	6	420
33. Figure 4.11 (f)	8,908	$Dist_{\vec{n}}$	11x11	9	431
34. Figure 4.12	3,230	$Dist_{\vec{n}}$	15x15	3	69
35. Figure 4.13	3,470	$Dist_{\vec{n}}$	15x15	3	87
36. Figure 4.14	4,960	$Dist_{\vec{n}}$	11x11	3	136
37. Figure 4.15	5,200	$Dist_{\vec{n}}$	11x11	3	126
38. Bunny's Leg	3,100	$Dist_{\vec{n}}$	11x11	3	52
39. Bunny's back	2,260	$Dist_{\vec{n}}$	11x11	3	35
40. Armadillo	4,090	$Dist_{\vec{n}}$	13x13	3	116
41. Ajax	2,670	$Dist_{\vec{n}}$	13x13	3	50

The experiments are done on an Intel Pentium 2.4GHz with 2.00 GB of RAM running Windows 7 OS. The computation time is dominant by the relief transferring part. The other steps of the algorithm contribute less than ten percent of all the computation time. From the Table 4.1, the complexity of the algorithm is quadratic with the number of vertices. The running time also scales linearly with the number of sampling points used for the Laplacian surface signature. However, increasing k-nearest neighbor value does not introduce much performance lost since the k-nearest neighbor can be query in sub-linear time. Typically, the k-nearest neighbor query has logarithmic complexity. But with small value of k and some optimization of the library, the query may have O(1) complexity. The types of distance metrics do not have effect on the computational time.

CHAPTER V

SUMMARY AND FUTURE WORK

5.1 Summary

This research proposes an algorithm to fill the hole on non-smooth surfaces using the available surface context. The algorithm can handle surface with relief patterns such as near-regular patterns, irregular patterns and stochastic patterns. This work would have a great benefit to surface acquisition process. The users do not need to spend a great amount of time repairing the scanned models. Another strong point of the proposed method is that no modification is done on the original surface. The preservation of the surface characteristics is crucial in application such as archiving ancient objects.

The main ideas of the algorithm are to decompose the input models into coarse mesh and relief mesh. The hole of the coarse mesh is filled using hole triangulation algorithm that minimize surface area and normal variation. The hole of the relief mesh is filled using example-base synthesis framework. The algorithm extracts the relief pattern from the surrounding surface and transfers it to the smoothly filled hole. In this work, the new surface signature is defined using Laplacian coordinates. Laplacian coordinate explicitly represents the local geometrical properties of shapes thus making it suitable for surface similarity test. Coarse mesh and relief mesh are combined in Laplacian space and then reconstruct using Inverse Laplacian transform.

The algorithm is tested with planar surfaces and curve surfaces that have near-regular patterns, irregular patterns and stochastic patterns. It is interesting to note that there are some connections between the characteristics of the normal vector component and the mean curvature component for points on the surface. Thus, one of these components may be sufficient to use as a surface signature's distance metric. When computing the Laplacian signature, the neighborhood size plays a dominant role in capturing the relief patterns. For near-regular relief patterns, the neighborhood size of at least 13x13 is required to capture the surface details. However, for stochastic patterns, the neighborhood size of 7x7 is usually enough. The number of nearest neighbor used to compute the Laplacian coordinate of a sampling point does not have

obvious influence on the quality of the results. Thus, the nearest neighbor size of three is adequate.

5.2 Future work

The quality of the coarse mesh can determine the quality of the result filled surface. If the mesh is not smooth enough, the relief pattern may not be clearly detectable. In addition, if the height of the relief pattern is too low, the Laplacian representation of the relief mesh may contain too much noise. This situation complicates the algorithm to detect the pattern of the surface. The author would like to explore geometric metrics that can be used to guide the smoothing process. Surface may not have to be smoothed evenly and isotropically. Some points of the surface may need more smoothing that the others.

Another interesting area of further research is on coarse mesh completion. The minimum area and normal variation are fine for smooth mesh completion. However, some characteristics of surface structures need additional geometric knowledge. Multi-scale curvature analysis, the curvature that define with more than 1-ring of neighborhood, may be useful to capture the structure of the model.

Most of the time the results from scanning devices contain surface fragments. It will be very useful to extend the proposed method to handle hole with isles. To handle hole with isles, many parts of the algorithm require modification. Active contour based methods may be more suitable for smooth hole completion than minimum surface area method. The available surface information of the fragments can be used as guidance when transferring relief pattern from the offset region. The surface fragments can also be used as anchor points when computing the inverse Laplacian transform. However, working with these fragments can introduce many difficulties. For example, each isle can have its own holes. Laplacian coordinates computed from the isles boundary are not accurate. It may be easier to design the algorithm if some loss in geometric detail of these fragments are acceptable.

REFERENCES

[1] Liepa, P. Filling holes in meshes. Eurographics SGP 2003 (2003): 200 – 205.

[2] Davis, J.; Marschner, S. R.; Garr, M.; and Levoy; M. Filling holes in complex surfaces using volumetric diffusion. 3DPVT 2002 (2002): 428–861.

[3] Sharf, A.; Lewiner, T.; Shklarski, G.; Toledo, S.; and Cohen, D. Interactive topology-aware surface reconstruction. Proceedings of ACM SIGGRAPH 2007 (2007): 431-439.

[4] Xie, H.; Mcdonell, K.; and Qin, H. Surface Reconstruction of Noisy and Defective Data Sets. Visualization 2004 (2004): 259-266.

[5] Schnabel, R.; Degener, P.; and Klein, R. Completion and Reconstruction with Primitive Shapes. Eurographics 2009 (2009): 503-512.

[6] Desbrun, M.; and Pottmann, H. Example-based 3D Scan Completion. Eurographic symposium on Geometry Processsing 2005 (2005).

[7] Breckon, T.; and Fisher, R. Non-Parametric 3D Surface Completion. 3D digital imaging and modeling 2005 (2005): 537-580.

[8] Sharf, A.; Alexa, M.; and Cohen, D. Context based surface completion. ACM Trans. on Graph 23 (2004): 878–887.

[9] Bendels, G.; Schnabel, R.; and Klein, R. Detail-preserving surface inpainting. VAST 2005 (2005):41-48.

[10] Park, S.; Guo, X.; Shin, H.; and Qin, H. Surface completion for shape and appearance. Visual Computing 2006 (2006): 168-180.

[11] Breckon, T.; and Fisher, R. Three-Dimensional Relief Surface Completion via Nonparametric Techniques. IEEE Transaction on Pattern Analysis and Machine Intelligence 2008 (2008): 2249-2255.

[12] Breckon, T.; and Fisher, R. A Hierarchical Extension to 3D Non-parametric Surface Relief Completion. Technical Report 2012 (2012).

[13] Sumner, R. W.; and Popovi´c, J. Deformation transfer for triangle meshes. Proceedings of ACM SIGGRAPH 2004 (2004): 399–405.

[14] Botsch, M.; Sumner, R.; Pauly, M.; and Gross, M. Deformation transfer for detail-preserving surface editing. In Proceedings of Vision, Modeling, and Visualization (VMV) 2006 (2006): 357–364.

[15] Turk, G. Texture synthesis on surfaces. Proceedings of ACM SIGGRAPH 2001 (2001): 347– 354.

[16] Alexei, A.; and Efros, Thomas K. L. Texture synthesis by non-parametric sampling. IEEE International Conference on Computer Vision 1999 (1999): 1033–1038.

[17] Wei, L.; and Levoy, M. Fast texture synthesis using tree structured vector quantization. Proceedings of ACM SIGGRAPH 2000 (2000): 479–488.

[18] Lorensen, W. E.; and Cline, H.E. Marching cubes: a high resolution 3D surface construction algorithm. Proceedings of ACM SIGGRAPH 1987 (1987): 163–170.

[19] Ju, T.; Lasasso, F.; Schaefer, S.; and Warren, J. Dual contouring of hermite data. Proceedings of ACM SIGGRAPH 2002 (2002): 339–346.

[20] Zorin, D.; Schr¨oder, P.; DeRose, T.; Kobbelt, L.; Levin, A.; and Sweldens, W. Subdivision for modeling and animation. Course notes of ACM SIGGRAPH 2000 (2000).

[21] Klincsek, G. Minimal triangulation of polygonal domains. Annals of Discrete Mathemathics 1980 (1980): 121–123.

[22] Barequet, G.; and Sharir, M. Filling gaps in the boundary of a polyhedron. Computer-Aided Geometric Design 1995 (1995): 207–229.

[23] Kettner, L. Using generic programming for designing a data structure for polyhedral surfaces. In 14th Annual ACM Symp. on Computational Geometry 14 (1998).

[24] Mantyla, M. An introduction to solid modeling. US: Computer Science Press, 1988.

[25] Carmo, M.P. Differential Geometry of Curves and Surfaces. US: Prentice Hall, 1976.

[26] Fiedler, M.. Algebraic connectivity of graphs. Czech. Math. Journal 23 (1973): 298–305.

[27] Meyer, M.; Desbrun, M.; Schr¨oder, P.; and Barr A.H. Discrete differential-geometry operators for triangulated 2- manifolds. In Visualization and Mathematics III (2003): 35–57.

[28] Sorkine, O. Differential Representations for Mesh Processing. Computer Graphics Forum 2006 (2006): 789–807.

[29] Lipman, Y.; Sorkine, O.; Cohen, D.; Levin, D.; Rossel, C.; and Seidel, H. Differential coordinates for interactive mesh editing. Shape Modeling International 2004 (2004): 181–190.

[30] George, A.; and Liu, J.W.H. Computer solution of large sparse positive definite matrices. US: Prentice Hall, 1981.

[31] George, A.; and Liu, J.W.H. The evolution of the minimum degree ordering algorithm. SIAM Review 1989 (1989): 1–19.

[32] Toledo, S.; Chen, D.; and Rotkin, V. Taucs: A library of sparse linear solvers. [Online]. 2003. Available from: http://www.tau.ac.il/stoledo/taucs [2012, March 1]

[33] Alliez, P.; Ucelli, G.; Gotsman, C.; and Attene, M. Recent advances in remeshing of surfaces. State-of-the-art report 2005 (2005).

[34] Pfeifle, R.; and Seidel, H.P. Triangular B-Splines for Blending and Filling of Polygonal Holes. Proc. Graphics Interface 1996 (1996): 186-193.

[35] Kobbelt, L.; Campagna, S.; Vorsatz, J.; and Seidel, H.P. Interactive Multi-Resolution Modeling on Arbitrary Meshes. Proceedings of ACM SIGGRAPH 1998 (1998).

[36] Moreton, H.P.; and S´equin, C.H. Functional optimization for fair surface design. Proceedings of ACM SIGGRAPH 1992 (1992): 167–176.

[37] Liang, L.; Liu, C.; Xu, Y.; Guo, B.; and Shum, H-Y. Real-time texture synthesis using patch-based sampling. ACM Transactions on Graphics 2001 (2001).

[38] Praun, E.; Finkelstein, A.; and Hugues Hoppe, H. Lapped textures. In Proceedings of ACM SIGGRAPH 2000 (2000): 465–470.

[39] Alexei, A.; Efros, T.; William T.; and Freeman, T. Image quilting for texture synthesis and transfer. In Proceedings of ACM SIGGRAPH 2001 (2001): 341–346.

[40] Kwatra, V.; Schodl, A.; Essa, I.; Turk, G.; and Bobick, A. Graphcut textures: image and video synthesis using graph cuts. ACM Trans. Graph 2003 (2003): 277–286.

[41] Bertalmío, M.; Sapiro, G.; Caselles, V.; and Ballester, C. Image Inpainting. Proceedings of ACM SIGGRAPH 2000 (2000).

[42] Pablo, A.; Gabriele, F.; Vicent, C.; and Guillermo, S. A Variational Framework for Exemplar-Based Image Inpainting. International Journal of Computer Vision 2011 (2011): 319-347.

[43] Kwatra, V.; Lefebvre, S.; Turk, G.; and Wei, L.Y. Example-based Texture Synthesis. SIGGPRAPH 2007 Course Notes 2007 (2007).

[44] Ying, L.; Hertzmann, A.; Biermann, H.; and Zorin, D. Texture and Shape Synthesis on Surfaces. Eurographics Workshop on Rendering 2001 (2001): 301–312.

[45] Zelinika, S.; and Garland, M. Interactive Texture Synthesis on Surfaces Using Jump Maps. Eurographics Symposium on Rendering 2003 (2003).

[46] Pravin, B.; Ingram, S.; and Turk, G. Geometric Texture Synthesis by Example. Eurographics/ACM Symposium on Geometry Processing 2004 (2004): 43–46.

[47] Zhou, K.; Xin, H.; Xi, W.; Yiying, T.; Mathieu, D.; Baining, G.; and Shum, H-Y. Mesh Quilting for Geometric Texture Synthesis. Proceedings of ACM SIGGRAPH 2006 (2006): 690–697.

[48] Sapidis, N.S. Designing Fair Curves and Surfaces: Shape Quality in Geometric Modeling and Computer-Aided Design. SIAM 1994 (1994).

[49] Botsch, M.; Pauly, M.; Kobbelt, L.; Alliez, P.; Levy, B.; Bischoff, S.; and Rossl, C. Geometric Modeling Based on Polygonal Meshes. SIGGRAPH 2006 Course Notes 2006 (2006).

[50] Desbrun, M.; Meyer, M.; Schr¨oder, P.; and Barr, A.H. Implicit fairing of irregular meshes using diffusion and curvature flow. In Proc. of ACM SIGGRAPH 1999 (1999): 317–324.

[51] Taubin, G. A signal processing approach to fair surface design. Proceedings of ACM SIGGRAPH 1995 (1995): 351–358.

[52] Perona, P.; and Malik, J. Scale-space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence 1990 (1990): 629–639.

[53] Fleishman, S.; Drori, I.; and Cohen, D.O. Bilateral mesh denoising. ACM Transactions on Graphics 2003 (2003): 950–953.

[54] Jones, T.R.; Durand, F.; and Desbrun, M. Non-iterative, feature-preserving mesh smoothing. ACM Transactions on Graphics 2003 (2003): 943–949.

[55] Floater, M.S.; and Hormann, K. Surface Parameterization: a Tutorial and Survey. In Advances in Multiresolution for Geometric Modelling 2005 (2005).

[56] Tutte, W. Convex representation of graphs. In Proc. London Math. Soc 1960 (1960).

[57] Floater, M. S.; Kos, G.; and Reimers, M. Mean value coordinates in 3d. CAGD 2005 (2005).

[58] Hormann, K.; and Greiner, G. MIPS: An efficient global parametrization method. Curve and Surface Design 2000 (2000): 153–162.

[59] Cohen-Steiner, D.; Alliez, P.; and Desbrun, M. Variational shape approximation. In Proc. of ACM SIGGRAPH 2004 (2004): 905–914.

[60] Khodakovsky, A.; Litke, N.; and Schr¨oder, P. Globally smooth parameterizations with low distortion. ACM TOG SIGGRAPH 2003 (2003).

[61] Zorin, D.; Schr¨oder, P.; and Sweldens, W. Interactive multiresolution mesh editing. In Proc. of ACM SIGGRAPH 1997 (1997): 259–268.

[62] Guskov, I.; Sweldens, W.; and Schr¨oder, P. Multiresolution signal processing for meshes. In Proc. of ACM SIGGRAPH 1999 (1999): 325–334.

[63] Kobbelt, F. OpenMesh Library. [Online]. 2012. Available from : http://www.openmesh.org [2012, March 1].

[64] Duff, I.; Grimes, R.; and Lewis, J. Sparse matrix test problems. ACM Trans. Math. Soft 1989 (1989): 1-14.

[65] Mount, D.M.; and Arya, S. ANN: A Library for Approximate Nearest Neighbor Searching. [Online]. 2012. Available from: http://www.cs.umd.edu/~mount/ANN/ [2010, March 1].

[66] Silva, S. PolyMeCo: Polygonal Mesh Analysis and Comparison Tool. [Online]. 2012. Available from: http://www.ieeta.pt/polymeco/indewx.php [2012, March 1].

BIOGRAPHY

Mr. Pongsagon Vichitvejpaisal was born on December 27, 1982 in Bangkok. He studied at Suankularb Wittayalai school. He graduated from Computer Engineering, Chulalongkorn University in 2004. He then earned master degree in Computer Engineering from Chulalongkorn University in 2006. He received H.M. the King's 72nd Birthday Scholarship and The Royal Golden Jubilee Ph.D Program Scholarship.