


Template Based Mesh Fitting Through a Set of Curves

CHOI, Yuet Kei



A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in

Automation and Computer-Aided Engineering

© The Chinese University of Hong Kong

February 2007

The Chinese University of Hong Kong holds the copyright of this thesis. Any person(s) intending to use a part or whole of the materials in the thesis in a proposed publication must seek copyright release from the Dean of Graduate School.



Thesis/ Assessment Committee

Professor DU, Ruxu (Chair)

Professor HUI, Kin Chuen (Thesis Supervisor)

Professor WANG, Changling Charlie (Committee Member)

Professor MA, Weiyin (External Examiner)

Acknowledgement

I would like to thank my Professor Kin-Chuen Hui, my research supervisor, who give me advise and guidance over the past research period. He also gives me a chance for my personal thinking such as organizing my research and independent work. His conscientiousness and full support lead me towards the success of the research

In addition, my appreciation goes to the colleagues in the Computer-Aided Design Laboratory for their technical support during my research period. Finally, I would like to thank for the Department of Automation and Computer-Aided Engineering of the Chinese University of Hong Kong for providing me a good working environment and advanced facilities throughout my research.

Abstract

Despite the large amount of work on mesh generation, the construction of a mesh with partitions matching a set of given curves remains a difficult process. Popular techniques used for mesh generation include the Delaunay Triangulation [1-3] and the Advancing Front method [4-6]. The Delaunay Triangulation operates by inserting a vertex satisfying the Delaunay criterion [7] in space. The determination of the location for the insertion of vertex is a complicated process, especially in 3D space. The Delaunay Triangulation approach is thus computational expensive. Moreover, the size of the mesh is uncontrollable and fairly large.

The Advancing Front method accepts a boundary mesh as input. The method begins by dividing the boundaries of the mesh into edges. Triangular faces are then generated one-by-one, starting from the boundary of the mesh, until the center of the region is reached. However, the initial division of boundaries and the overlapping problem of the mesh polygons are complicated.

In spite of the substantial amount of work on the generation of mesh, the construction of mesh with partitions has not been well addressed. In this thesis, an approach for the generation of mesh with partitions is presented. In the proposed method, a mesh with partitions is created by fitting a template mesh to a set of curves. The mesh created by our technique is free of self-overlapping of the mesh polygons and the mesh can be used in computer graphics and computer-aided design applications.

To fit a template mesh to a set of curves, a global deformation technique is employed. The global deformation technique includes a coarse-level and a fine-level

deformation. The coarse-level global deformation adopts the Radial Basis Functions (RBFs) to adjust the orientation and the shape of the template mesh. Then, the fine-level global deformation uses the closest point method to determine the correspondences between the template meshes and the curves. The RBFs is then applied to adjust the mesh to interpolate the set of curves. However, some faces are flipped after the deformation. In this thesis, a face flip prevention technique is proposed.

As the topology of the template mesh is not affected in the global deformation, the deformed template mesh may not match the partitions as depicted by the set of curves. To create partitions of the mesh, the polygon mesh in the vicinity of the curves is re-triangulated. Finally, the mesh is smoothed by using the mean-curvature and the Laplacian flow methods. A technique to improve the mean-curvature method is also presented. The improved mean curvature technique is used to remove the possible distortion of the mesh. The Laplacian flow method is applied to create a more evenly distributed mesh. Experimental results show that the technique generates the required mesh with partitions effectively.

摘要

儘管在網格產生的技術上有大量的研究，要使生成的網格符合一組曲線仍然是一個艱難的工作。比較流行的網格生成技術包括三角剖分法和推進陣面法。三角剖分法的原理是透過在空間內加入一些符合三角法則的點，以生成新的面。但是，用這方法去計算一個新點的位置是很複雜的，特別是在三維空間裡。故此，三角剖分法需要大量的電腦計算資源。除此之外，網格內面的數量是不可控制。

在推進陣面法裡，主要的輸入為一個有邊界的網格。推進陣面法的原理是透過切開網格的邊界，再由邊界慢慢生成面，最後到網格的中心。但是，邊界切開的算法和生成面之間的重疊問題是很複雜的。

雖然有大量的工作研究網格的生成，但是生成一個有不同區域的網格卻很少被討論。這論文建議了一個技術去生成不同區域的網格。在這技術中，網格是由一個模板網格和一組曲線組成。生成的模板不但沒有網格內部重疊的問題，而且生成的網格能應用於在電腦繪圖與自動化電腦設計應用上。

要使一個模板網格符合一組曲線，需用上範圍的變形技術，這範圍的變形技術包含了粗糙及細緻的變形。在粗糙的變形中，輻射基底函數應用於調整模板網格的形狀及它的位置。當模板網格經過粗糙的變形後，下一步是細緻的變形。在細緻的變形中，最近點的搜尋方法用於建立模板網格與曲線之間的對應點。當模板網格與曲線之間的對應點建立後，輻射基底函數會再次應用於用於模板網格上，使模板網格變形符合一組曲線。但是，當模板網格經過變形後，模板網

格上的面可能會有不正確的變形效果。在這論文中，我們建議了一種技術去防止這種不正確的變形。

模板網格在變形時，它的結構並不會受影響。故此，變形後的模板網格可能不符合一組曲線所描述的分割部份。要生成一個有不同分割部份的模板網格，曲線附近的部份網格必須經過三角剖分。最後，平均曲率運算法則及拉普拉斯運算法則會應用於模板網格上，使它的外形變得平滑。另外，我們提議了一種改善平均曲率運算法則的技術。這修改了的平均曲率運算法則會用於消除模板網格上可能存在的扭面，而拉普拉斯運算法則則用於生成一個分佈更平均的模板網格。實驗顯示我們提議的技術，能有效地生成一個有不同分割部份的網格。

Contents

Acknowledgement	i
Abstract	ii
Contents.....	vi
List of Figures	viii
1. Introduction	1
1.1 Previous Works	3
1.1.1 Template deformation	3
1.1.2 Mesh partitioning	3
1.1.3 Mesh Smoothing	4
1.2 Overview of the approach.....	5
1.3 Thesis outline	7
2. Global Deformation.....	8
2.1 The closet point method	10
2.1.1 Computational complexity	11
2.2 Deformation Techniques	12
2.2.1 Existing deformation method.....	12
2.2.2 Radial Basis Functions (RBFs).....	14
2.2.2.1 Computational complexity	18
2.2.3 Result	18
2.3 Face flip prevention.....	20
2.3.1 Detection of the flipped face	22
2.3.1.1 Common approach:	22
2.3.1.2 Our Approach.....	23
2.3.1.3 Comparisons of the face flip detection method:	26
2.3.2 Local Subdivision	27
3. Partitioning of the mesh.....	29
3.1 Existing method	29
3.2 Our approach.....	31
3.3 Computational complexity	34
4. Mesh smoothing algorithm.....	35
4.1 The Laplacian flow method.....	36
4.2 The mean-curvature method.....	40
4.3 Our Approach.....	43
4.3.1 The modified mean-curvature method.....	43
4.3.2 The modified Laplacian flow method.....	45
4.3.3 Feature constraints.....	47

4.3.4 Computational complexity	47
4.4 Comparison of the mesh smoothing approach	48
5. Implementation and Results.....	51
5.1 Construction of the template mesh and boundary curves.....	51
5.2 Selection of the corresponding vertex pairs.....	52
5.3 Results	54
6. Conclusions	63
6.1 Future development	65
Appendix A	66
Determination of the projected path on a mesh:	66
Reference.....	68

List of Figures

Figure 1.1: The template mesh before (left) and after (right) global deformation	2
Figure 1.2: The template mesh before (left) and after (right) mesh partitioning.....	2
Figure 1.3: The template mesh before (left) and after (right) mesh smoothing	2
Figure 2.1: The flow chart of the deformation process	9
Figure 2.2: Feature point.....	10
Figure 2.3: Example of determining the correspondence	11
Figure 2.4: The cup template and the curves before deformation. The red and blue labels are the corresponding vertex pairs.....	19
Figure 2.5: The cup deformed by using the bi-harmonic basis function (left). The cup deformed by using the tri-harmonic basis function (right).	19
Figure 2.6: The template mesh before deformation (left) and after deformation (right)	21
Figure 2.7: The face before deformation (left). The face after deformation (right) ...	23
Figure 2.8: The face flip detection process.....	25
Figure 2.9: The general face flip detection process.....	26
Figure 2.10: Local face subdivision.....	28
Figure 2.11: Deformation of the subdivided face.....	28
Figure 2.12: Deformation of the subdivided face.....	28
Figure 3.1: Constructing a curve on a mesh using the Dijkstra's algorithm.....	30
Figure 3.2: Li's approach.....	30
Figure 3.3: Face inserted in the clear-out region	32
Figure 3.4: Process of re-triangulating the mesh.....	32
Figure 3.5: The undetected region on the boundary	33
Figure 3.6: Treatment on the mesh boundary	33
Figure 4.1: The Laplacian flow method	36
Figure 4.2: Original sphere	38
Figure 4.3: Distortions on the sphere (top). Sphere after 20 iterations of the Laplacian flow method is applied (bottom).....	38
Figure 4.4: Original torus	39
Figure 4.5: Distortion on the torus (top). Torus after 20 iterations of the Laplacian flow method is applied (bottom).....	39
Figure 4.6: Angle α_i and β_i are used to estimate the mean curvature vector.....	40
Figure 4.7: Original sphere	41
Figure 4.8: Distortion on the sphere (top). Sphere after 60 iterations of the mean- curvature method is applied (bottom).....	41
Figure 4.9: Original torus	42

Figure 4.10: Distortion on the torus (top). Torus after 60 iterations of the mean-curvature method is applied (bottom).....	42
Figure 4.11: Our mean-curvature method	44
Figure 4.12: Our Laplacian flow method	46
Figure 4.13: Smoothing a deformed sphere.....	49
Original sphere (top). Distortions on the sphere (middle left). Sphere after 20 iterations of the Laplacian flow method (middle right), 60 iterations of the mean-curvature method (bottom left) and 100 iterations of our approach (bottom right) is applied.....	49
Figure 4.14: Smoothing a deformed torus	50
Original Torus (top). Distortions on the torus (middle left). Torus after 20 iterations of the Laplacian flow method (middle right), 60 iterations of the mean-curvature method (bottom left) and 100 iterations of our approach (bottom right) is applied	50
Figure 5.1: The template mesh with a set of corresponding vertices (left). The template mesh with incorrectly selected corresponding vertices (middle). The curves with the corresponding vertices (right).	53
Figure 5.2: The deformed template mesh with a proper set of corresponding vertex pairs (left), and the deformed template mesh with improper corresponding vertex pairs (right)	53
Figure 5.3: The shoe template and the curves network (top). 7 corresponding point pairs are specified (bottom left). The shoe template after the coarse-level global deformation is applied (bottom right).....	54
Figure 5.4: The shoe template before the fine-level global deformation is applied (left). The shoe template after the fine-level global deformation is applied (right).	55
Figure 5.5: The shoe template before (top) and after (bottom) partitioning.....	55
Figure 5.6: The shoe template before smoothing (left). The shoe template after applying the modified mean-curvature method (right).....	56
Figure 5.7: The shoe template before smoothing (left). The shoe template after applying the modified Laplacian flow method (right).....	56
Figure 5.8: Final mesh with partitions.....	56
Figure 5.9: Results of using different template mesh	57
Figure 5.10: Results of using fewer feature constraints	58
Figure 5.11: The effect of feature constraints. Result with more feature constraints (left). Result with fewer feature constraints (right).	58
Figure 5.12: The elephant template before deformation (top left). The elephant template after deformation (top right). Final mesh (bottom).....	59
Figure 5.13: Using an open template mesh to fit a set of curves.....	60

Figure 5.14: Results of using template mesh with different resolutions. The template mesh before deformation (left).The template mesh after deformation (middle).

Final mesh (right). 60

Figure 5.15: Fitting a template mesh to different set of curves (1). 61

Figure 5.16: Fitting a template mesh to different set of curves (2) 62

Figure A.1: Determination of the projected path on the mesh. 67

1. Introduction

Mesh generation is commonly used in computer graphics and computer-aided design applications. However, given a set of curves, it is difficult to generate a mesh with partitions matching the set of curves. Popular techniques used for mesh generation include the Delaunay Triangulation [1-3] and the Advancing Front method [4-6]. However, the size of the mesh generated by the Delaunay Triangulation and the Advancing Front methods is uncontrollable. The generated mesh is especially unsuitable for applications involving the use of the Finite Element (FEM) and Boundary Element Method (BEM). This thesis addresses the problems by fitting a template mesh to a set of curves

The proposed method in this thesis accepts a set of curves and a template mesh as inputs. Our goal is to create a mesh with partitions defined by the set of curves. By associating correspondences between the curves and the template mesh, the template mesh is deformed to interpolate the curves. Our method consists of three stages: 1) global deformation, 2) mesh partitioning and 3) mesh smoothing. In the first stage, correspondences between the curves and the template mesh are established. Then, the template mesh is deformed to interpolate the curves by adjusting the orientation and the shape of the template mesh (Figure 1.1). To create partitions of the mesh, a technique for re-triangulating the mesh is proposed and is applied in the second stage (Figure 1.2). In the final stage, the mesh is smoothed by using the mean-curvature and the Laplacian flow methods. The mean curvature method is used to remove the distortion of the mesh and the Laplacian flow method is applied to create a more evenly distributed mesh (Figure 1.3).

Since the size of generated mesh by our proposed method is depended on the size of inputted mesh, the size of the mesh is controllable and can be used in the Finite Element Analysis (FEM) and Boundary Element Analysis (BEM)

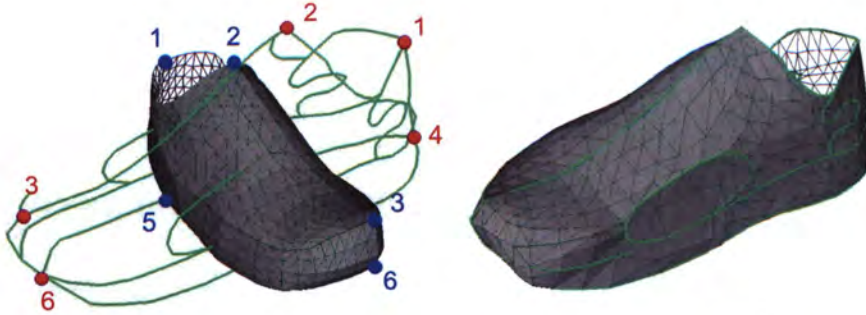


Figure 1.1: The template mesh before (left) and after (right) global deformation

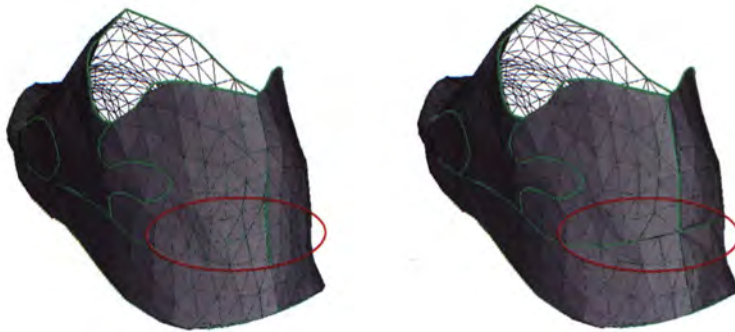


Figure 1.2: The template mesh before (left) and after (right) mesh partitioning

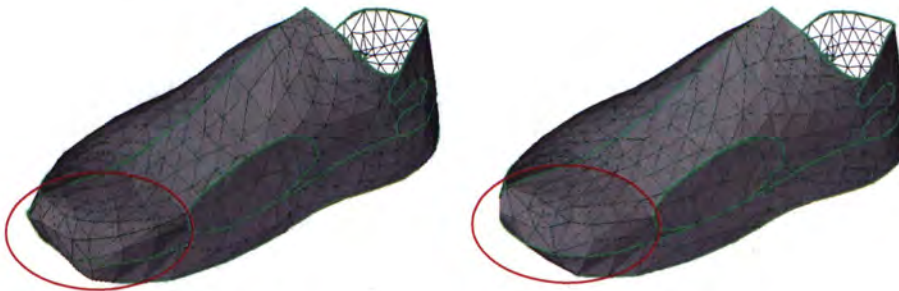


Figure 1.3: The template mesh before (left) and after (right) mesh smoothing

1.1 Previous Works

Related works can be classified into three main categories. They are the deformation of template mesh, the partitioning of polygon mesh, and the smoothing of polygon mesh.

1.1.1 Template deformation

Global Deformation using a set of corresponding vertices has been widely studied [8-11]. Kahler [8] proposed to deform a template head model by fitting the template mesh to a scanned model iteratively. However, the methods proposed in [8, 9, 10] failed to map the template to the scanned head model if there are holes on the surface of the scanned model. Kahler et al. [11] solved the problems by fitting a template mesh to the scanned model using the Radial Basis Functions (RBFs).

1.1.2 Mesh partitioning

Creating partitions on the mesh is particularly important in the application of 3D morphing. Kanai [30] proposed to specify regions on a mesh by constructing curves on the mesh. The curves are considered as boundaries of the mesh partitions. Given two vertices on the model specifying the start and end points of the boundary curve, Kanai and Suzuki [12] adopted the Dijkstra's algorithm to determine the boundary curves by locating the shortest path along the edges of the mesh. However, global searching of the edges of the mesh is required and the accuracy of the path relies very much on the resolution of the mesh. Mitchel et al. [13] and Kapoor [14] adopted the

wavefront propagation method to compute the shortest path. Chen and Han [15] proposed an algorithm to find the shortest path by unfolding the faces of the mesh. However, the methods in [13, 14, 15] are computationally expensive and are not suitable for mesh with a high resolution. Li [31] proposed an algorithm similar to that of Kanai and Suzuki [12]. Additional vertices are required to partition the mesh. Despite the accuracy of the path on the mesh is improved, the mesh size is largely increased after the mesh is partitioned.

1.1.3 Mesh Smoothing

Mesh smoothing is the process for reducing distortions on an object. A major difficulty of mesh smoothing is to remove the distortion while preserving the desirable features of the object. A common way for smoothing a polygon mesh is to use the Laplacian flow method [16, 17]. Despite the Laplacian flow method can create a more evenly distributed mesh, the shape of the mesh is altered and the mesh is shrunk after smoothing. Taubin [18] proposed to use two scale factors with opposite sign to suppress the shrinkage of the mesh. However, the shape of the mesh is altered by the smoothing process.

Another approach for smoothing a polygon mesh is based on the mean-curvature flow method [19, 20]. Desbrum [19] smooth a polygon mesh by computing the mean curvature of the mesh vertices. Ohtake et al. [21] proposed a technique to improve the mean-curvature flow method. Despite the method is good for smoothing polygon mesh, irregular polygons on the mesh is not reduced.

1.2 Overview of the approach

Our approach accepts a set of curves and a template mesh as inputs. The goal is to create a mesh covering regions partitioned by the set of curves. Based on the inputs, a set of corresponding vertex pairs between the curves and the template mesh are defined in advance. In order to simplify our investigation, we assume

1. the curves are connected by linear segments;
2. the template mesh is composed of triangular elements; and
3. the corresponding vertex pairs between the curves and the template that used to specify the orientation between the template mesh and the curves are available.

The proposed method consists of three stages: 1) global deformation, 2) mesh partitioning and 3) smoothing. In the first stage, the template mesh is deformed using the Radial Basis Functions (RBFs). Different from the method proposed by Kahler et al [11], our method deforms the template mesh in two steps. The first step is the coarse-level deformation. Based on the given corresponding vertex pairs, the orientation and the shape of the template mesh are adjusted. The second step is the fine-level global deformation which is used to fit the template mesh to the curves. Another set of correspondences between the template mesh and the curves are determined to fit the template mesh to the curves. In this step, the closest point method is adopted. The closest point method computes the distance between the vertices that defines the curves and the template mesh. Then, all the vertices that define the curves are associated with the closest vertices on the template mesh. The vertices on the template mesh are translated to the corresponding vertices on the

curve. However, some faces may be flipped after deformation. Thus, a face flip prevention technique is proposed to solve this problem.

In the second stage, the partitions of the mesh are created. Instead of using the algorithm proposed by Li [31] and Kanai [12], our method creates the partitions of the mesh by re-triangulating the mesh in the vicinity of the curves. During the re-triangulation process of the mesh, the topology of the mesh is modified locally. The re-triangulation process is repeated until the shortest path between two given vertices on the mesh is formed. After the re-triangulation process, some distortions are usually induced on the mesh. Therefore, a smoothing technique similar to that of Ohtake [23] is proposed to remove the distortion and make the mesh more evenly distributed. In our technique, the mean curvature and the Laplacian flow methods are adopted. A technique to improve the mean-curvature method is also presented. The improved mean-curvature technique is used to remove the distortion of the mesh. The Laplacian flow method is applied to improve the distribution of the mesh. Different from the Laplacian flow method presented in [22], our technique focuses on the improving the distribution of the mesh and reducing shrinkage rather than removing the distortion on the mesh.

Our approach not only creates a mesh with partitions in different resolutions, the possible self-overlapping of the mesh polygons is eliminated. In addition, there is no significant change in the size of the mesh. The mesh with partitions created using our approach is particularly useful in computer graphics and computer-aided design applications.

1.3 Thesis outline

The structure of the thesis is organized as follows. In this chapter, a review of the related works and the overview of our approach are presented. In chapter 2, the method of Radial Basis Functions (RBFs) and the problems resulting from the deformation are discussed. In chapter 3, the methods to create a mesh in regions partitioned by the curves are discussed. The mesh smoothing algorithm is presented in the chapter 4. Chapter 5 describes the implementation and the experimental results. Conclusions and future developments are given in chapter 6.

2. Global Deformation

This chapter discusses the global deformation technique that is used to fit the template mesh to the curve. Given a set of corresponding vertex pairs between the curves and the template mesh, the global deformation technique is applied to adjust the orientation and the shape of the template mesh such that it interpolates the curves. The method deforms the template mesh in two steps. The first step is the coarse-level deformation. Based on the given corresponding vertex pairs, the orientation and the shape of the template mesh are adjusted using RBFs. The second step is the fine-level global deformation which is used to fit the template mesh to the curves. Another set of correspondences between the template mesh and the curves are determined. The closest point method is adopted to compute the distance between the vertices of the curves and the template mesh. Then, the RBFs is applied again to fit the template mesh to the curves. Figure 2.1 illustrates the flow chart of the deformation process.

In this chapter, the closest point method that is used to determine the correspondences between the template mesh and the curves is described in Section 2.1. The Radial Basis Functions (RBFs) and some experimental results using RBFs for object deformation are presented in Section 2.2. Finally, a technique that is used to prevent the possible reversion of the faces is given in Section 2.3.

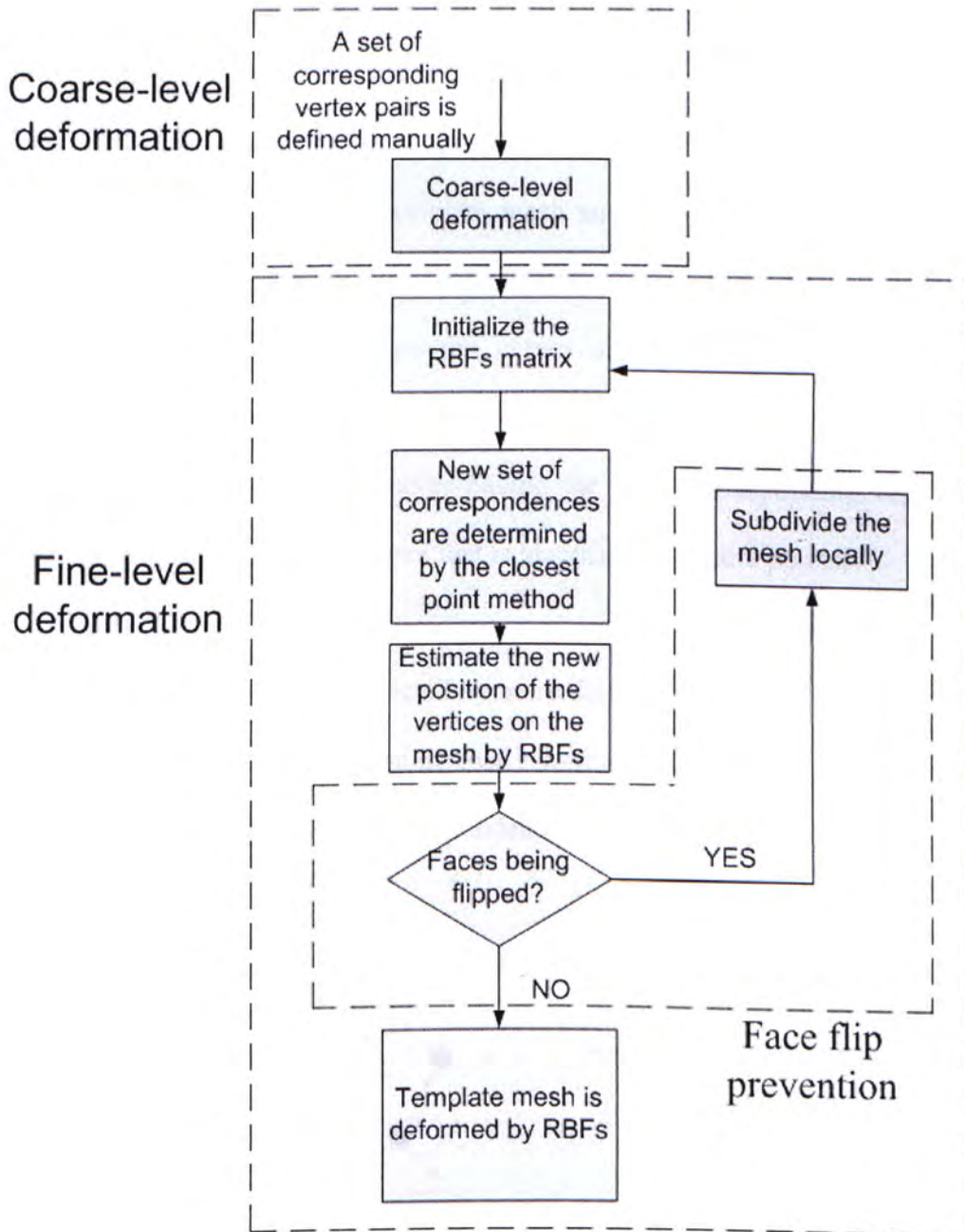


Figure 2.1: The flow chart of the deformation process

2.1 The closet point method

After the orientation and the shape of the template mesh are adjusted in the coarse-level deformation, the closet point method is applied to determine the correspondences between the template mesh and the curves. In this method, one vertex on the curve is associated with a vertex on the mesh. The correspondences between the vertices that define the curves and the vertices on the mesh are established by considering the shortest distance between the vertices. If there are more than one vertex on the curves having the same corresponding vertex on the mesh, priority is given to the vertex that is identified as feature points.

A feature point is a vertex connected by more than two linear segments (Figure 2.2). It specifies the point shared by more than two regions of the mesh, and thus has the highest priority in forming the correspondences with the template mesh.

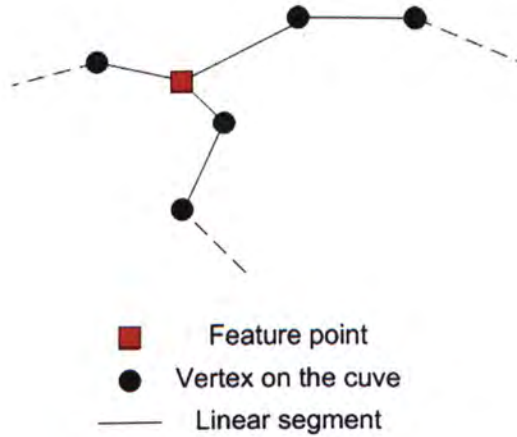


Figure 2.2: Feature point

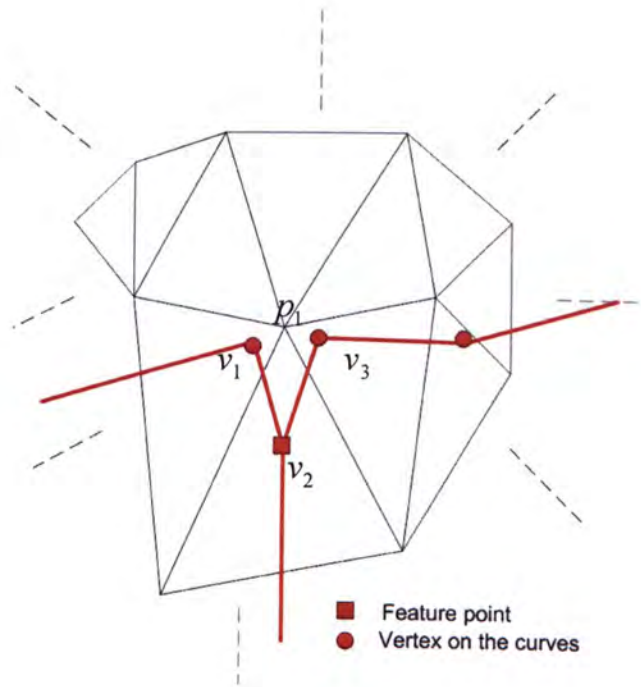


Figure 2.3: Example of determining the correspondence

Suppose p_1 is a vertex on the template mesh and v_1, v_2, v_3 are the vertices that define the curves. As v_2 is the intersection point of three linear segments, it is identified as the feature point. In Figure 2.3, the distance between p_1 and v_2 is the longest. Since v_2 is defined as the feature point, the corresponding vertex pair between the template mesh and the curve is p_1 and v_2 .

2.1.1 Computational complexity

Consider a set of curves with n number of vertices, and the template mesh contains m number of vertices. Since the closet point method is applied for all vertices on the curves, the time complexity is $O(mn)$.

2.2 Deformation Techniques

Given the correspondences between the template mesh and the curves, the next step is to deform the template mesh. In this section, the existing global deformation methods and the radial basis functions (RBFs) are presented

2.2.1 Existing deformation method

In the following, two existing approaches that used affine transformation for deformation are described.

a) For the first approach, the deformation is determined by evaluating and applying the affine transformation to the template iteratively [32]. The affine transformation can be found by the least squared method. Given a set of n corresponding vertices, which are represented by the matrices $\mathbf{A} \in R^{n \times 3}$ and $\mathbf{B} \in R^{n \times 3}$. The goal is to minimize the squared error $\|\mathbf{AX} - \mathbf{B}\|^2$ such that the mapping function \mathbf{X} can be determined.

The over-determined matrix equation is as follow:

$$\mathbf{AX} = \mathbf{B} \quad (2.1)$$

\mathbf{X} in (2.1) can be solved by using (2.2)

$$\mathbf{X} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{B} \quad (2.2)$$

where \mathbf{X} is an affine transformation.

Overview of the approach

Input: Corresponding vertices on the template mesh are stored in matrix **A**

Corresponding vertices on the curves are stored in matrix **B**

Output: Affine transformation matrix **X** with 16 degrees of freedom.

Since the maximum degree of freedom is only 16 for an affine transformation, the deformation effect is limited.

(b) In the second approach, the affine transformation is determined by [24] and is optimized by evaluating a set of error functions: marker error E_m and smoothness error E_s . The marker error E_m measures the distance between each corresponding vertex pair on the template mesh and the curves. The smoothness error E_s measures the change in deformation of the neighboring vertices over the template surface. The goal is to minimize the combined error $E = E_m + E_s$.

The marker error

$$E_m = \sum_{i=1}^m \| T_{k_i} v_{k_i} - m_i \|^2 \quad (2.3)$$

where T_{k_i} is the affine transformation of a vertex v_{k_i} on the template mesh and

m_i is the corresponding vertex on the curves.

The smoothness error

$$E_s = \sum_{\{i,j|\{v_i,v_j\} \in \text{edges}(M_T)\}} \| T_i - T_j \|^2 \quad (2.4)$$

where T_i is the affine transformation for the vertex v_i , T_j is T_i 's neighboring vertex and M_T is the template mesh.

Overview of the approach

Input: The vertex v_{k_i} on the template mesh and the vertex m_i on the curves

Output: Affine transformation T_i

2.2.2 Radial Basis Functions (RBFs)

In this thesis, the Radial Basis Functions (RBFs) is adopted for the global deformation. Comparing with those methods which use affine transformation, the degree of RBFs is dependent on the number of correspondences. Therefore, its deformation effect is better than the methods introduced in Section 2.2.1.

In the RBF global deformation technique, it accepts a set of curves, a template mesh with no off surface point and a set of corresponding vertex pairs between the curves and the template mesh. Let n be the number of the correspondences between the template mesh and the curves, $v_i \in R^3$ and $v'_i \in R^3$ be the i^{th} corresponding vertices on the template mesh and the curves respectively, where $i = 1, 2, \dots, n$. The Radial Basis Function (RBFs) determines the mapping function $E(v_i)$ that maps v_i to v'_i such that:

$$v'_i = E(v_i) \tag{2.5}$$

The mapping function is used to fit the template mesh to the curves. It is expressed with an elastic function which is a weighted linear combination of n basis functions ϕ . The mapping function is determined by the correspondences between the template mesh and the curves.

$$E(v_i) = \sum_{j=1}^n \alpha_j \phi(\|v_i - v_j\|) + \mathbf{R}v + \mathbf{t} \quad (2.6)$$

where $\alpha_i \in R^3$ are weights, $\mathbf{R} \in R^{3 \times 3}$ is the rotation matrix and $\mathbf{t} \in R^3$ is the translation vector. $\|v_i - v_j\|$ is the distance between the corresponding vertices.

In this thesis, the bi-harmonic basis function $\phi(r) = r$ is chosen to give the linear and global interpolating effect. In Equation (2.6), α_i, \mathbf{R} and \mathbf{t} are the unknown to be determined. The mapping function contains $3(n+4)$ unknowns to be determined, however there are n correspondences which give only $3n$ constraints. To provide enough constraints for the system, the following compatibility constraints are included:

$$\sum_{i=1}^n \alpha_i^k = \sum_{i=1}^n \alpha_i^k v_i^1 = \sum_{i=1}^n \alpha_i^k v_i^2 = \sum_{i=1}^n \alpha_i^k v_i^3 = 0 \quad (2.7)$$

where $k = 1, 2, 3$ and $v_i = \begin{bmatrix} v_i^1 \\ v_i^2 \\ v_i^3 \end{bmatrix}$.

To set up a system of linear equations related to the correspondences between the template mesh and the curves, $\phi(\|v_i - v_j\|)$ and \mathbf{R} in Equation (2.6) are denoted by

p_{ij} and $\begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix}$ respectively. Thus, Equation (2.5) can be expressed in matrix form:

$$\begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} & v_1^1 & v_1^2 & v_1^3 & 1 \\ P_{21} & P_{22} & \cdots & P_{2n} & v_2^1 & v_2^2 & v_2^3 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} & v_n^1 & v_n^2 & v_n^3 & 1 \\ v_1^1 & v_2^1 & \cdots & v_n^1 & 0 & 0 & 0 & 0 \\ v_1^2 & v_2^2 & \cdots & v_n^2 & 0 & 0 & 0 & 0 \\ v_1^3 & v_2^3 & \cdots & v_n^3 & 0 & 0 & 0 & 0 \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \\ R_1 \\ R_2 \\ R_3 \\ t \end{bmatrix} = \begin{bmatrix} v_1' \\ v_2' \\ \vdots \\ v_n' \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.8)$$

Or

$$\mathbf{KX} = \mathbf{Y} \quad (2.9)$$

The matrix \mathbf{K} is symmetric and positive definite unless all v_i are coplanar. When v_i are coplanar, additional corresponding vertices are required. Those vertices can be determined by v_i and its vertex normal n_i .

Since matrix \mathbf{K} is symmetric and positive definite, the solution of \mathbf{X} is unique [27]. The Equation (2.9) can be solved by the LU Decomposition method [28]. The mapping function can then be applied to the template mesh. Let m be the total

number of vertices on the template mesh. Denote $v_j^t = \begin{bmatrix} v_j^{t1} \\ v_j^{t2} \\ v_j^{t3} \end{bmatrix}$ as the j^{th} vertex on the

template mesh before deformation, where $j = 1, 2, \dots, m$, and express a basis function $q_{ji} = \phi(\|v_j^t - v_i\|)$. We can determine the updated position of the vertex

v_j^t on the template mesh by:

$$\begin{bmatrix} v_1' \\ v_2' \\ \vdots \\ v_m' \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} & v_1^{t1} & v_1^{t2} & v_1^{t3} & 1 \\ q_{21} & q_{22} & \cdots & q_{2n} & v_2^{t1} & v_2^{t2} & v_2^{t3} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} & v_m^{t1} & v_m^{t2} & v_m^{t3} & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \\ R_1 \\ R_2 \\ R_3 \\ t \end{bmatrix} \quad (2.10)$$

2.2.2.1 Computational complexity

The RBFs requires evaluating \mathbf{X} in Eq (2.9). A typical practice to solve Eq (2.9) is by using the LU Decomposition method [28] of which computational complexity is $O((n+4)^3)$. The time complexity for applying the mapping function to vertices on the template mesh by Eq (2.10) is $O(3m(n+4))$. Therefore, the total computational complexity for RBFs is $O((n+4)(3m+(n+4)^2))$

2.2.3 Result

This section compares the deformation results by using the bi-harmonic basis function $\phi(r) = r$ and the tri-harmonic basis function $\phi(r) = r^3$. The bi-harmonic basis function is used to give the linear interpolation effect and the tri-harmonic basis function gives a smooth interpolation effect by approximating the in-between correspondences. Since the tri-harmonic basis function is a spline function, the vertices will be non-linearly interpolated which may result in undesirable distortion of the mesh.

Figure 2.4 shows the template mesh and the curves before deformation. The red and blue labels are the given corresponding vertex pairs between the template mesh and the curves. Figure 2.5 shows the results of using the bi-harmonic and the tri-harmonic basis functions. As illustrated in Figure 2.5, the body and the ear part of the cup is distorted by using the tri-harmonic basis function.

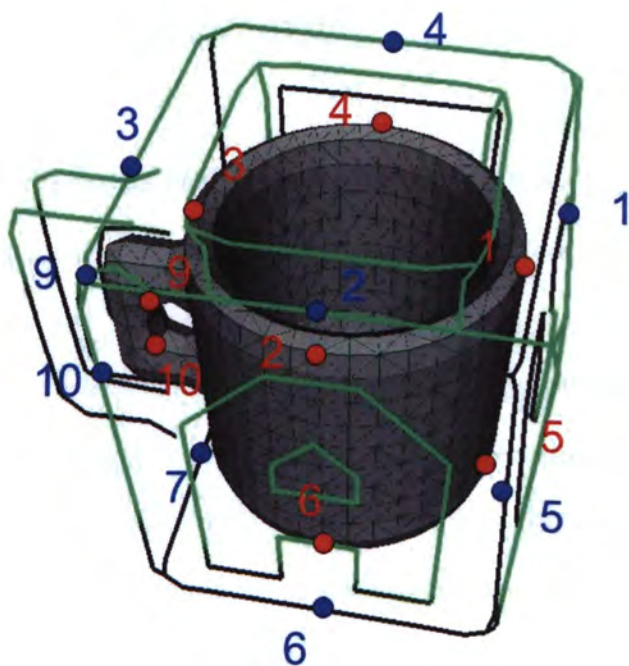


Figure 2.4: The cup template and the curves before deformation. The red and blue labels are the corresponding vertex pairs.

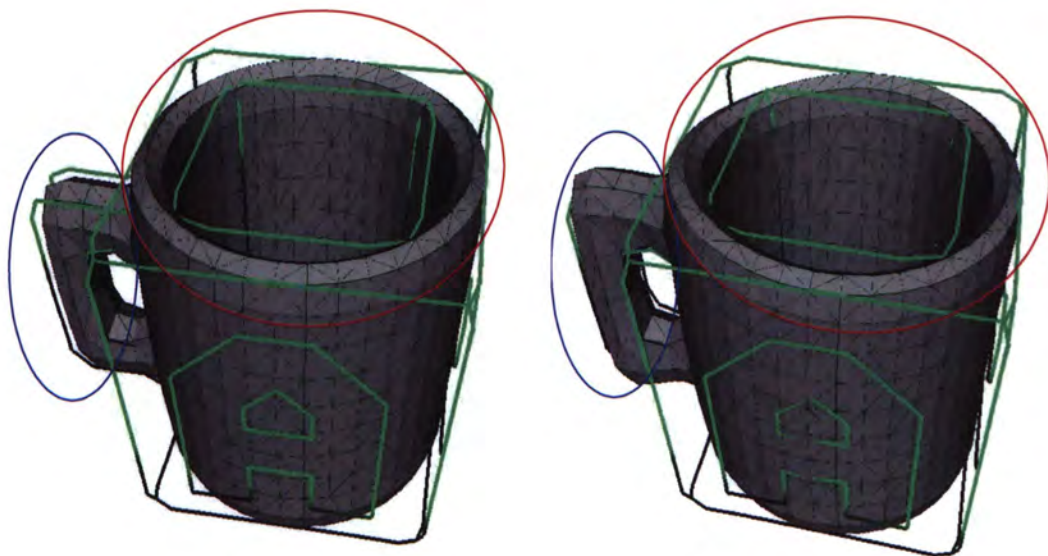


Figure 2.5: The cup deformed by using the bi-harmonic basis function (left). The cup deformed by using the tri-harmonic basis function (right).

2.3 Face flip prevention

The global deformation process may result in some faces being flipped. Figure 2.6 shows an example of face flip in a template mesh before and after deformation. In this example, the vertices on the template mesh is deformed based on the corresponding vertices on the curves, and is not affected by the deformation using the RBFs. Therefore, the face (blue color) may be flipped after the deformation. In this section, a face flip prevention technique is proposed. The technique consists of two major processes. The first process is the face flip detection process. In this process, the faces which may be flipped after deformation are identified. The second process is the local subdivision process. After the faces which may be flipped are identified, local subdivision is applied to prevent the flip of faces after deformation. The face flip detection and the local subdivision processes iterate until no flipped face is detected. In this section, existing methods and our proposed face flip detection method are presented in Section 2.3.1. Then, the subdivision process is described in Section 2.3.2.

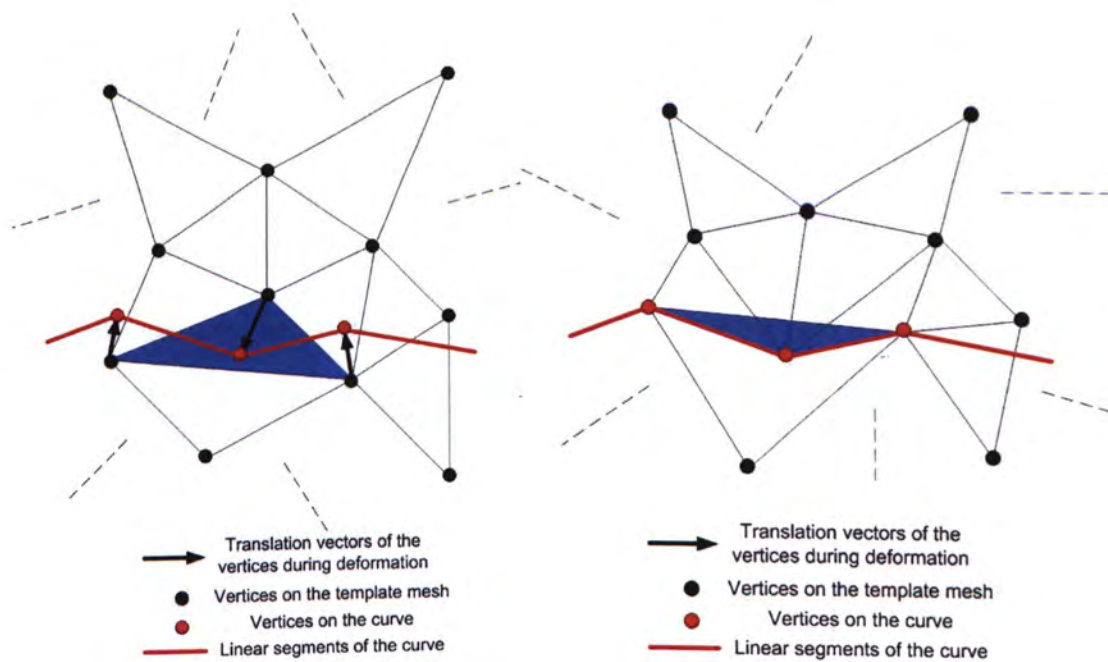


Figure 2.6: The template mesh before deformation (left) and after deformation (right)

2.3.1 Detection of the flipped face

In this section, a technique for detecting the flipped faces is proposed. Common methods for detecting the flipped faces are described in Section 2.3.1.1. Our technique detects the flipped faces by considering the variation of vertices location and is presented in Section 2.3.1.2.

2.3.1.1 Common approach:

In adjusting polygon vertices to match with the boundary curves, there are two possible cases for a face to be flipped.

Case 1: Planar deformation

Numerous methods can be used to detect the flipped face when the deformed face is coplanar with the un-deformed face. A common method is to examine the sign of the triangle's area. Denote the vertex of a triangle as $v_i = (x_i, y_i)$, the signed area Δ of a planar triangle is defined by:

$$\Delta = \frac{1}{2} \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

By determining the possible change in sign of Δ , the flipped face can be detected.

Case 2: Non-planar deformation

A common technique is to determine the changes in the face's normal before and after the deformation. A face is flipped if the angle between the face's normal before and after the deformation is greater than a specified tolerance.

2.3.1.2 Our Approach

Let p_i and p_i' be the i^{th} vertex of the face before and after the deformation respectively.

The vector between p_i and p_i' is defined as t_i and the vector between p_2 and p_3 is denoted as d_1 (Figure 2.7). A flip-counter is used to count the number of intersection between t_i and d_1 . The flip-counter is incremented by one if an intersection between t_i and d_1 is detected. This is repeated for each vertex of the same polygon. The final value of the flip-counter is then used for identifying if the triangular face has been flipped or not.

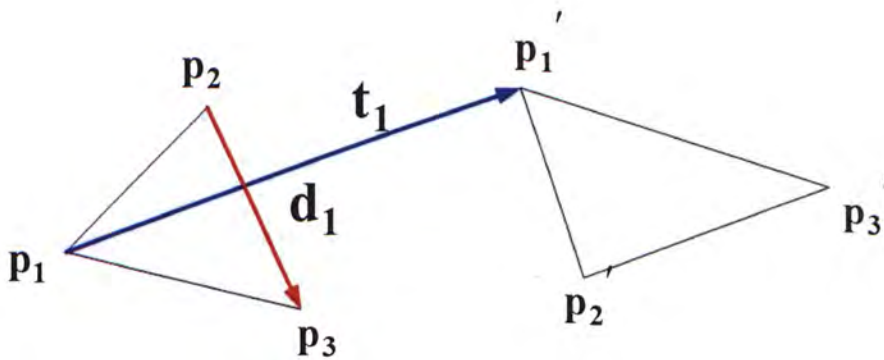


Figure 2.7: The face before deformation (left). The face after deformation (right)

Case 1: Planar deformation

In a deformation, the first vertex \mathbf{p}_1 moves to \mathbf{p}_1' , \mathbf{d}_1 is a vector defined as

$$\mathbf{d}_1 = s(\mathbf{p}_3 - \mathbf{p}_2)$$

where s is a constant. Define a vector \mathbf{t}_1 as

$$\mathbf{t}_1 = r(\mathbf{p}_1' - \mathbf{p}_1)$$

where r is a constant.

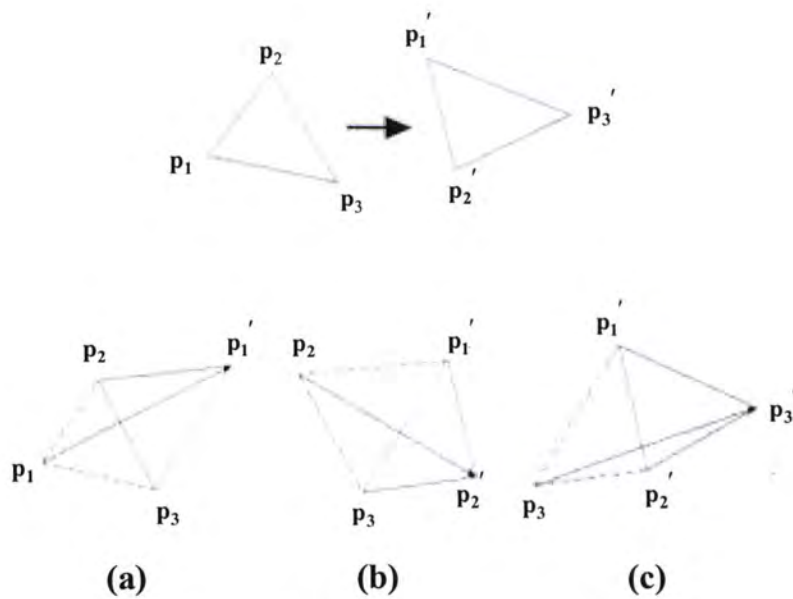
The intersection between \mathbf{t}_1 and \mathbf{d}_1 satisfies the following

$$\mathbf{p}_1 + \mathbf{t}_1 = \mathbf{p}_2 + \mathbf{d}_1$$

Hence

$$\mathbf{p}_1 + r(\mathbf{p}_1' - \mathbf{p}_1) = \mathbf{p}_2 + s(\mathbf{p}_3 - \mathbf{p}_2)$$

As \mathbf{d}_1 and \mathbf{t}_1 lie on the same plane, the intersection between \mathbf{d}_1 and \mathbf{t}_1 can be located by solving for r and s . If $0 \leq r \leq 1$, \mathbf{d}_1 intersects with \mathbf{t}_1 and the flip-counter is incremented by one. Figure 2.8 illustrates the process of detecting the flipped face. The process is applied to all vertices of the triangular face. The face is flipped if the final value of the flip-counter is odd.

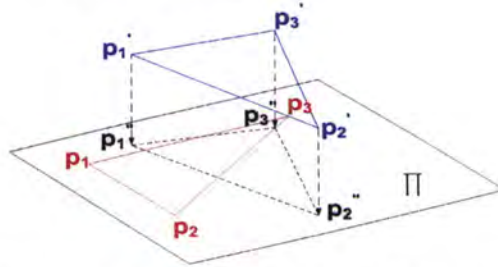


- (a) t_1 intersects with d_1 . The flip-counter is incremented by one.
- (b) t_2 intersects with d_2 . The flip-counter is incremented by one.
- (c) t_3 intersects with d_3 . The flip-counter is incremented by one.

Figure 2.8: The face flip detection process

Case 2: Non-planar deformation

In most cases, the vertices before and after the deformation lie on different planes. Thus, it may not be possible to detect the intersection between t_i and d_i directly. Instead, the vertices after the deformation are projected onto the plane Π where possible intersections are detected. The plane Π is defined by the vertices before the deformation as shown in Figure 2.9.



p_1 , p_2 and p_3 are the vertices of the triangle before deformation

p_1' , p_2' and p_3' are the vertices of the triangle after deformation

p_1'' , p_2'' and p_3'' are the projection of the vertices p_1' , p_2' and p_3' on the plane Π respectively

Figure 2.9: The general face flip detection process

Instead of using the vertices p_1' , p_2' and p_3' , the projected vertices p_1'' , p_2'' and p_3'' are used in the face flip detection process for planar deformation.

2.3.1.3 Comparisons of the face flip detection method:

In the non-planar case, a common technique to detect the flipped face is to determine the changes in the face's normal before and after the deformation. A face is flipped if the angle between the face's normal before and after the deformation is greater than a specified tolerance. In this method, the tolerance has to be carefully adjusted so that all flipped face can be detected. The method is inefficient as different tolerance may have to be used for different models. In our proposed face flip detection

method, t_i , d_i .and flip counter is adopted to detect the flipped face. Since t_i and d_i is detected automatically, our proposed method is automatically and more efficiency.

2.3.2 Local Subdivision

In the previous section, the face flip detection technique is proposed to identify the face which may be flipped after deformation. In this section, a face flip prevention technique is proposed.

For the faces which will be flipped in the deformation, local subdivision is applied to avoid face flip. The face flip detection and local subdivision processes iterate until no flipped face is detected. Figure 2.10 illustrates the result of subdividing a face. The closest point method is applied to determine a new set of correspondences between the curves and the template mesh. As shown in Figure 2.11 and 2.12, the subdivided face is not flipped after the deformation no matter p_1 or p_2 is determined as the corresponding vertex. Figure 2.11 illustrates the case when p_1 is determined as the corresponding vertex. Figure 2.12 illustrates the case when p_2 is determined as the corresponding vertex

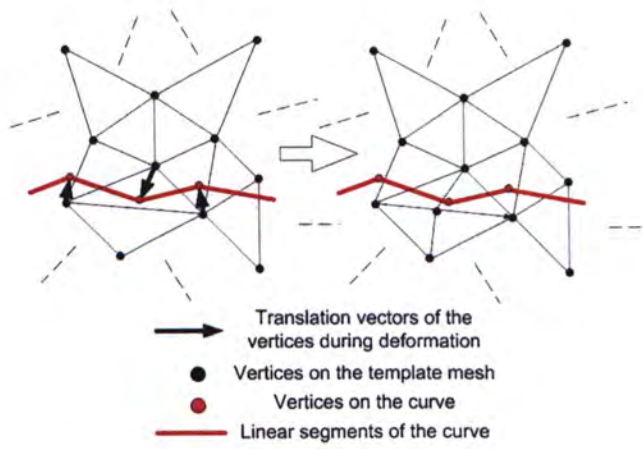


Figure 2.10: Local face subdivision

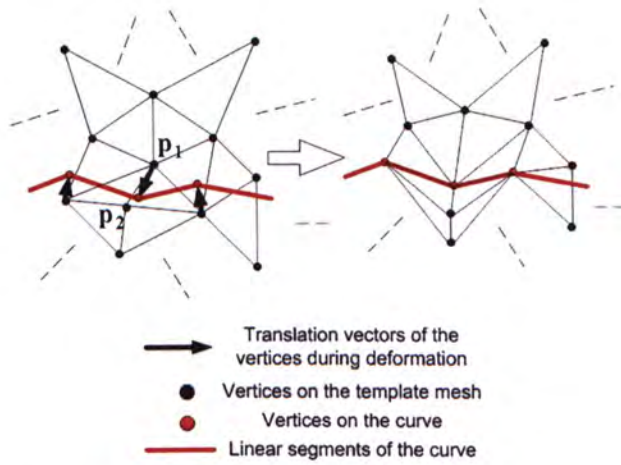


Figure 2.11: Deformation of the subdivided face

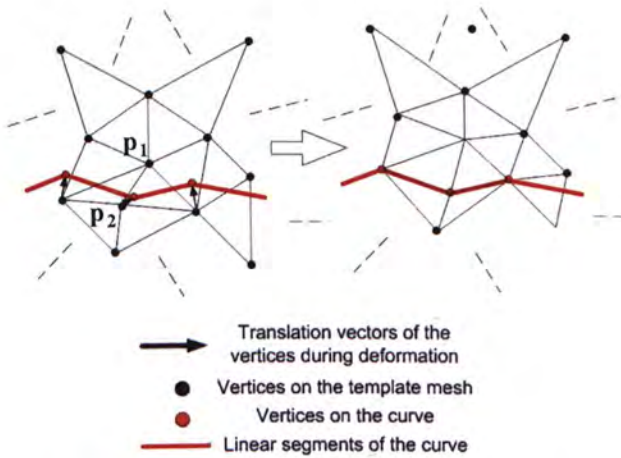


Figure 2.12: Deformation of the subdivided face

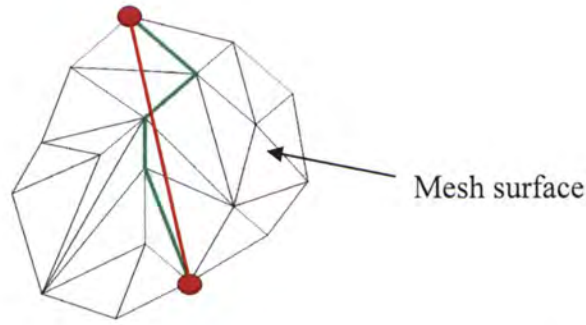
3. Partitioning of the mesh

After the global deformation, a template mesh is fitted to a set of curves. Then, the mesh has to be re-triangulated in regions partitioned by the curves. In this chapter, a technique for re-triangulating the mesh is proposed. Two existing methods and our approach for creating a mesh in regions partitioned by the curves are described in Section 3.1 and Section 3.2 respectively.

3.1 Existing method

Creating partitions on a mesh is particularly important in the application of 3D morphing. Kanai [30] proposed a technique to specify regions of a mesh by constructing a curve on the mesh. The curve is then used as the boundary curve between partitions of the mesh.

(a) To specify a curve on a mesh, a simple approach is to construct the curve along the edges of the mesh. It means that vertices on the curves are vertices on the mesh. Kanai and Suzuki [12] constructed the curves by linking two vertices of the curve with the shortest path over the mesh, and the Dijkstra's algorithm is adopted in their approach (Figure 3.1).

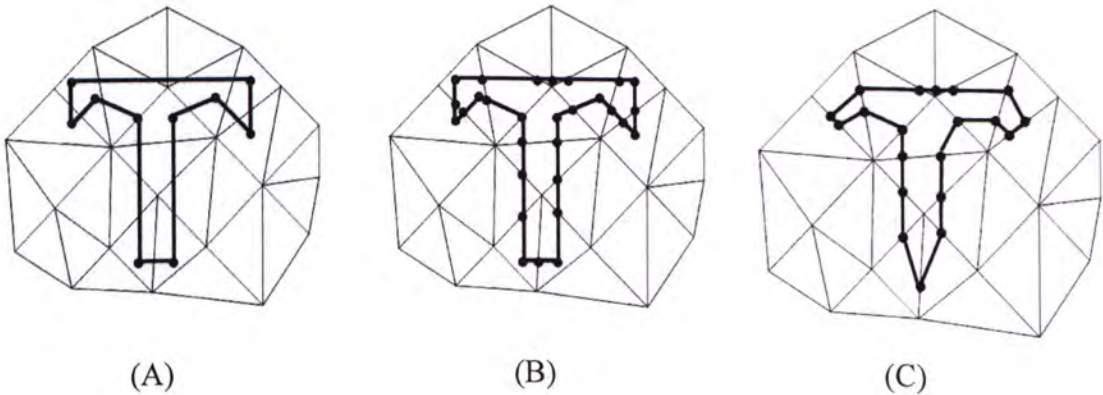


The path (red in color) represents the original curve

The path (green in color) represents the Dijkstra's shortest path

Figure 3.1: Constructing a curve on a mesh using the Dijkstra's algorithm

(b) Another approach to create a mesh in regions partitioned by the curves is proposed by Li [24]. In their approach, vertices are inserted on the mesh if there are intersections between the boundary of the face and the curve. However, this increases the mesh size of the object. Figure 3.2 illustrates Li's approach.



**A: A curve specified on a mesh; B: Vertices are inserted on the mesh edges;
C: Vertices on the curves are removed (Final result)**

Figure 3.2: Li's approach

3.2 Our approach

Given the vertices that define the linear segment of the curves, the mesh in the vicinity of the linear curve segment (clear-out region) is re-triangulated. The clear-out region is the region created by the faces which the projected path of the linear curve segment lies on. The clear-out region is detected by projecting the linear segment of the curve onto the mesh. The direction of projection is determined based on the direction of the face's normal. By locating a projected path of the linear segment on the mesh, the clear-out region can be detected. The method of determining a projected path on the mesh is described in Appendix A.

To re-triangulate the mesh in the clear-out region, the faces are removed first. Then, an edge which is the same as the linear segment of the curve is inserted. The remaining faces are inserted according to the size of the angles of the clear-out region. In Figure 3.3a, suppose A_4 is the smallest angle between the edges in the clear-out region. New face is then inserted in the clear-out region as shown in Figure 3.3b. Figure 3.3c shows the result after the re-triangulation process. The process of re-triangulating a mesh is shown in Figure 3.4. Since the same number of faces is inserted in the clear-out region, the size of the mesh is not affected in the re-triangulation process.

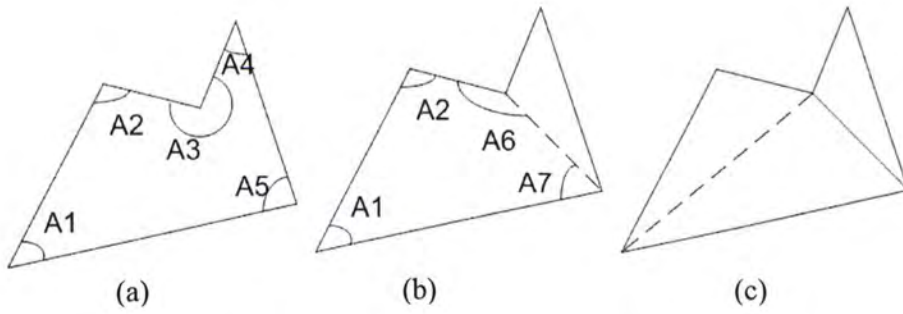
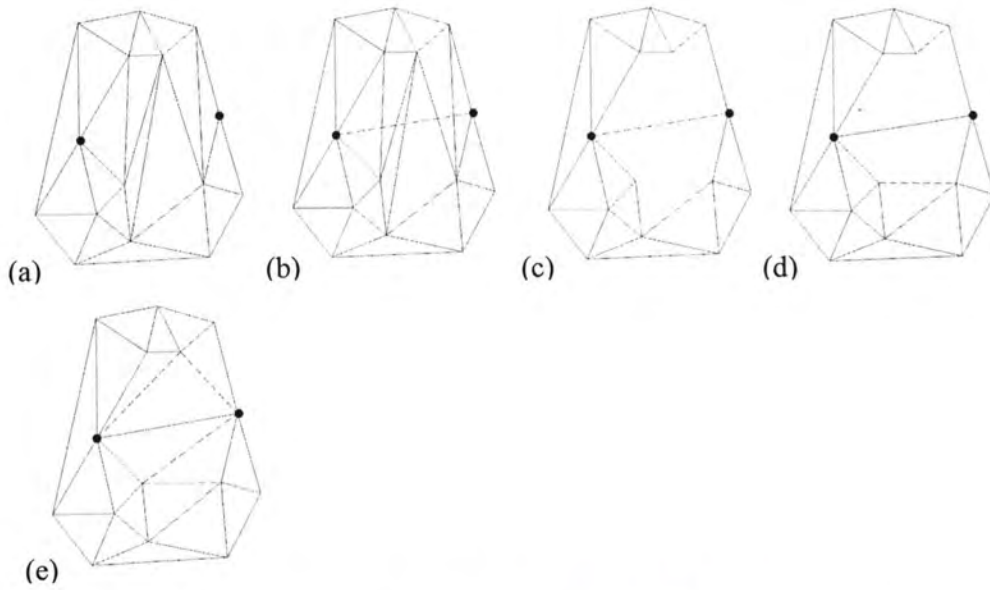


Figure 3.3: Face inserted in the clear-out region

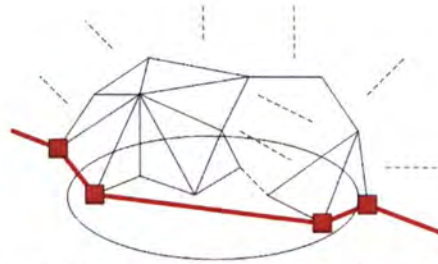


- (a) Given a mesh and two vertices that define the curve
- (b) New edge is inserted.
- (c) Remove the faces in the clear-out region.
- (d) Insert the faces in the cleared-out region
- (e) Result

Figure 3.4: Process of re-triangulating the mesh

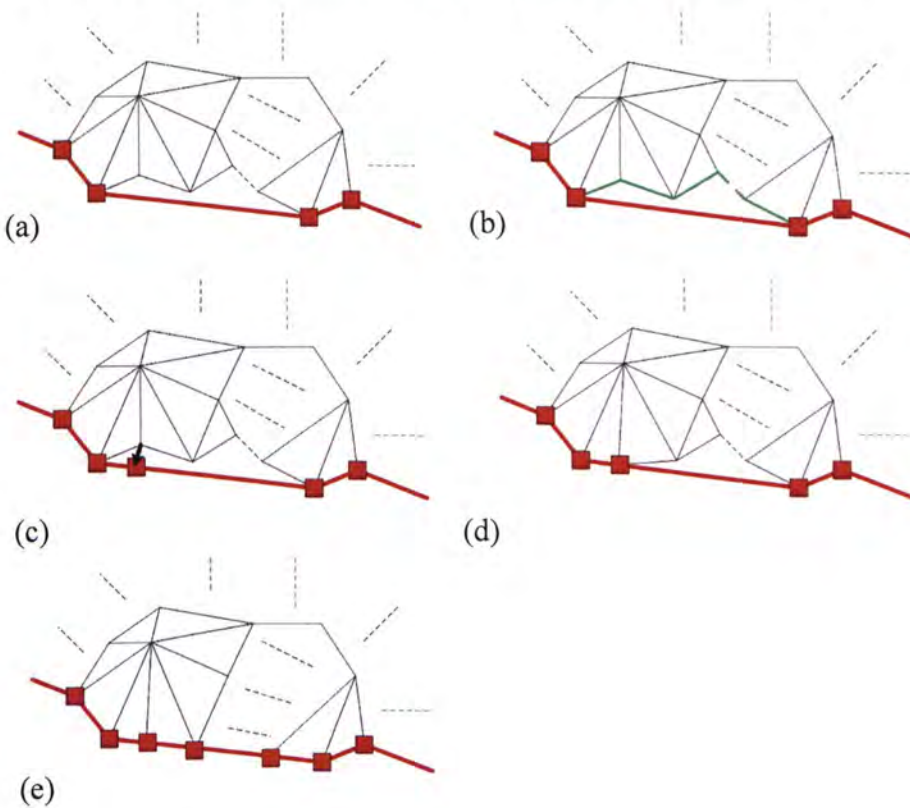
Since the clear-out region is located by projecting the inserted edge on the mesh, the clear-out region may be undetected for some concave regions, especially on the boundary of the mesh (Figure 3.5). In this case, the Dijkstra's algorithm is adopted to

detect the shortest path between two vertices on the mesh. The vertex on the Dijkstra's path is then moved to the inserted edge by projection. The process is repeated until the shortest path between two vertices on the mesh is obtained (Figure 3.6). As the topology of the mesh is modified, some distortion may be induced on the mesh



Clear-Out region that cannot be detected
by using our method

Figure 3.5: The undetected region on the boundary



- (a) Initial mesh;
- (b) Dijkstra's path is detected on the mesh;
- (c) – (d) The vertices are moved to the inserted edge;
- (e) Result;

Figure 3.6: Treatment on the mesh boundary

3.3 Computational complexity

Assume the template mesh contains m number of faces and n number of vertices respectively. Since a global search for all faces is required in locating the clear-out region, the computational complexity is $O(m^m)$. For the undetected region on the mesh, the shortest path between two vertices is detected by using the Dijkstra's algorithm of which computational complexity is $O(n^2)$. Therefore, the total computational complexity is $O(n^2 + m^m)$.

4. Mesh smoothing algorithm

After the template mesh is partitioned by the set of curves, some distortions are usually induced on the mesh. The distribution of the mesh may also be affected in the deformation. This chapter proposes a technique to remove the distortion on the mesh and to produce a mesh with more evenly distributed triangular faces.

Mesh smoothing is a process to remove distortions from the object. The objective of the mesh smoothing process is to remove the distortions while preserving the desired features of the object. A popular method to smooth a mesh is to use the Laplacian flow [16,17] and the mean-curvature methods [18]. In this chapter, the proposed smoothing technique is discussed. The Laplacian flow method is described in Section 4.1. The mean-curvature method is discussed in Section 4.2. Our approach to smooth the mesh is presented in Section 4.3. Finally, experimental results on mesh smoothing using our approach and a comparison with the Laplacian flow and mean-curvature methods are shown in Section 4.4.

4.1 The Laplacian flow method

Let p^k be a vertex on the mesh after the k^{th} iteration. The new position of p^k in the next iteration is defined by

$$p^{k+1} = p^k + \lambda \Delta(p^k) \quad (4.1)$$

where p^{k+1} is the new position of p^k , $\Delta(p^k)$ is the displacement vector of p^k and λ is a positive constant defined by the user

The Laplacian flow method computes p^{k+1} by determining the displacement vector $\Delta(p^k)$ which is defined by the umbrella-operator [29]:

$$\Delta(p^k) = \frac{1}{\sum_{i=1}^m w_i} \sum_{i=1}^m w_i q_i - p^k \quad (4.2)$$

where q_i is the i^{th} neighboring vertex of p^k (one-ring vertex of p^k), m is the total number of neighboring vertices of p^k (See Figure 4.1) and w_i is the weighting of q_i . The mesh is smoothed by computing p^{k+1} using Equation (4.1) repeatedly.

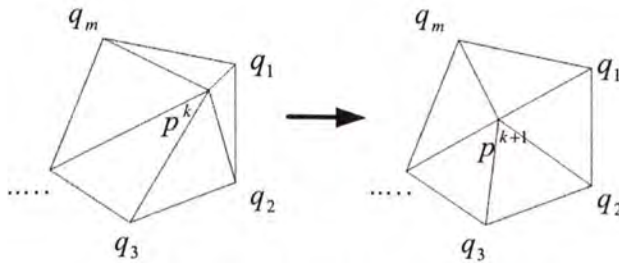


Figure 4.1: The Laplacian flow method

The weight w_i can be chosen in many different ways. The most common choice is to set $w_i = 1$ such that Equation (4.2) becomes:

$$\Delta(p^k) = \frac{1}{m} \sum_{i=1}^m q_i - p^k \quad (4.3)$$

Another choice of w_i is to compute the inverse of the distance between p^k and q_i by:

$$w_i = \|p^k - q_i\|^{-1} \quad (4.4)$$

J. Vollmer [22] proposed to include the previous vertex p^k in the calculation.

Equation (4.1) becomes:

$$p^{k+1} = \alpha p^k + \frac{1-\alpha}{\sum_{i=0}^m w_i} \sum_{i=0}^m w_i q_i \quad (4.5)$$

where α is a constant, $0 \leq \alpha \leq 1$. However, the shrinkage problem cannot be solved.

Another approach is to include the vertex p^0 in the calculation [22]. Equation (4.5)

becomes:

$$p^{k+1} = \alpha p^0 + \frac{1-\alpha}{\sum_{i=0}^m w_i} \sum_{i=0}^m w_i q_i \quad (4.6)$$

Despite the Laplacian flow method can create a more evenly distributed mesh, the shape of the mesh is altered and the mesh is shrunk after smoothing. Figure 4.2 and Figure 4.4 show the mesh of a sphere and torus before smoothing. Figure 4.3 and Figure 4.5 show the results of the meshes after applying the Laplacian flow method (Equation 4.3). Figure 4.3 shows that the sphere is shrunk after using the Laplacian flow method. As illustrated in Figure 4.5, the torus is also shrunk after using the Laplacian flow method.

Results

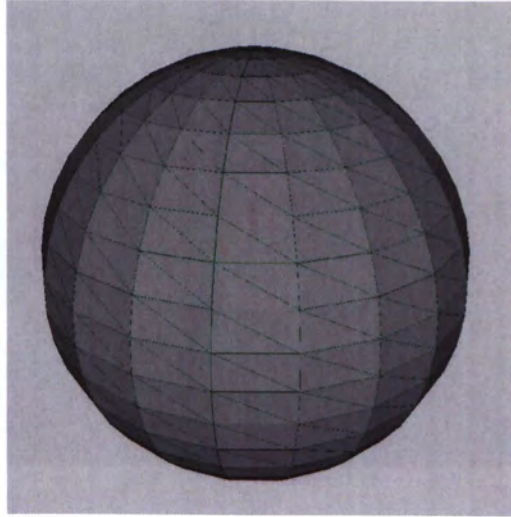


Figure 4.2: Original sphere

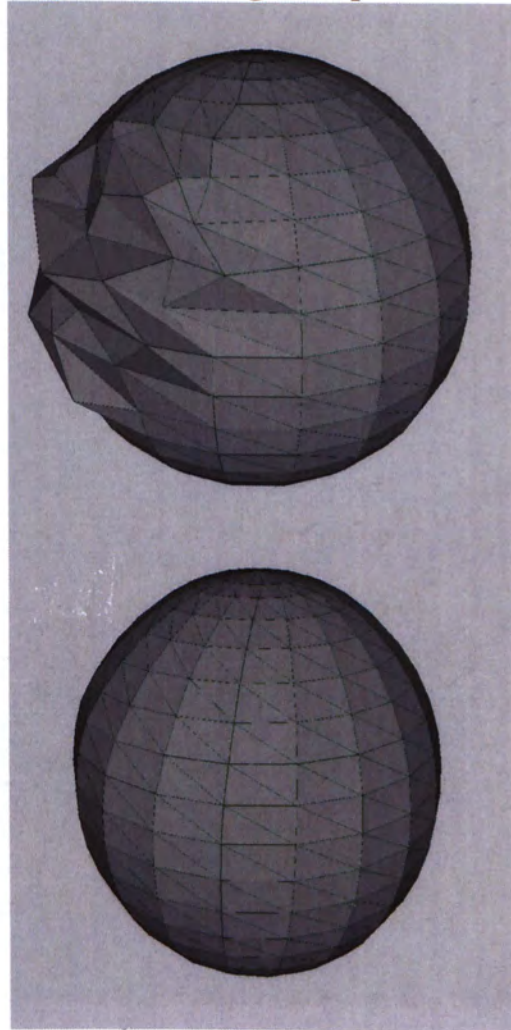


Figure 4.3: Distortions on the sphere (top). Sphere after 20 iterations of the Laplacian flow method is applied, where $\lambda = 1$ (bottom).

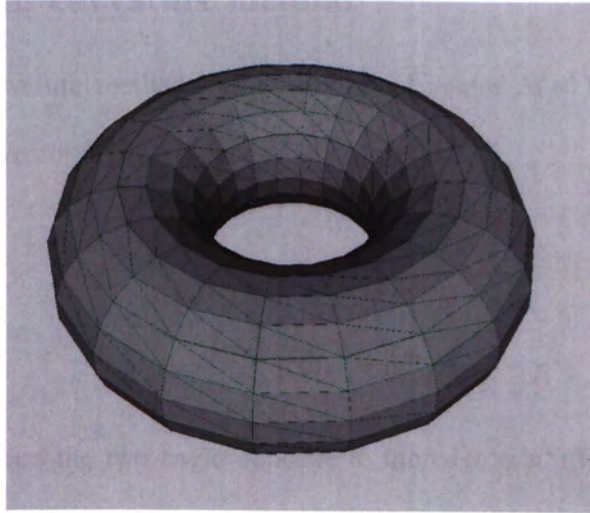


Figure 4.4: Original torus

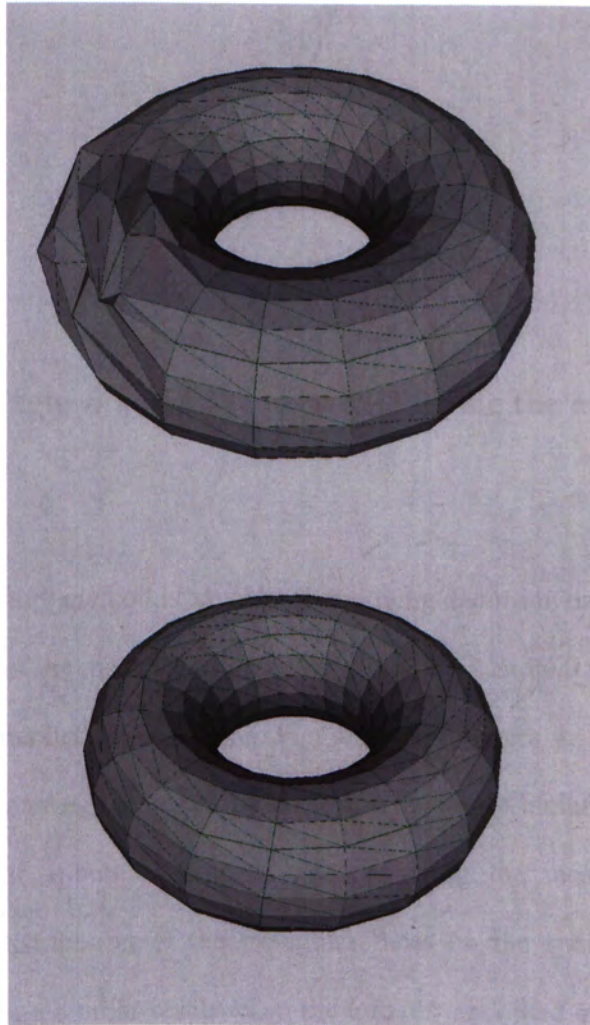


Figure 4.5: Distortion on the torus (top). Torus after 20 iterations of the Laplacian flow method is applied, where $\lambda = 1$ (bottom).

4.2 The mean-curvature method

In the mean-curvature method, the displacement vector $\Delta(p^k)$ is replaced by the mean curvature vector $H(p^k)$. Equation (4.1) becomes:

$$p^{k+1} = p^k + \lambda H(p^k) \quad (4.7)$$

with

$$H(p^k) = \frac{1}{4A} \sum_{i=0}^m (\cot \alpha_i + \cot \beta_i)(q_i - p^k) \quad (4.8)$$

where α_i and β_i are the two angle opposite to the edge $q_i p^k$ (Figure 4.6), and A is the sum of areas of the triangles surrounding p^k .

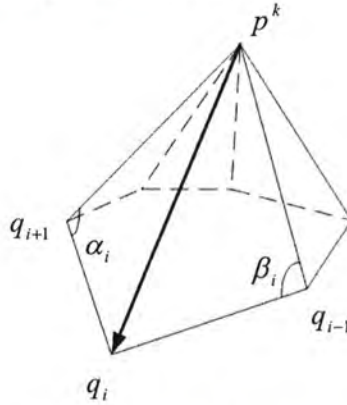


Figure 4.6: Angle α_i and β_i are used to estimate the mean curvature vector

The mean-curvature method is capable of removing distortion on the mesh. However, the distribution of the mesh is not improved. Figure 4.7 and 4.9 show the meshes of a sphere and a torus before smoothing. Figure 4.8 and Figure 4.10 show the results of the meshes after being smoothed by the mean curvature method. In Figure 4.8, the distortion on the sphere is reduced after applying the mean-curvature method. However, the distribution of the triangular faces on the sphere is not improved. Figure 4.9 shows a similar result when the torus is smoothed by the mean-curvature method.

Results

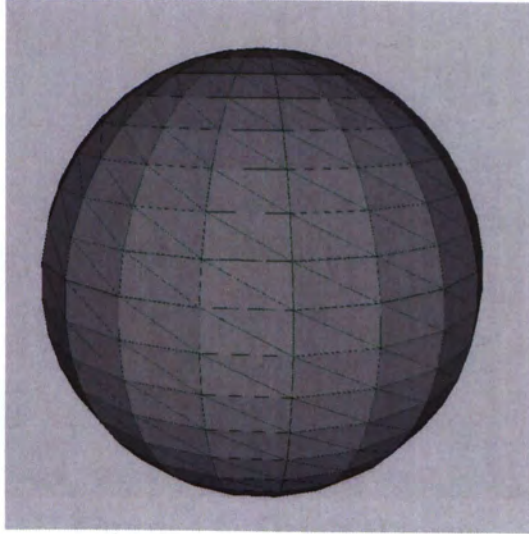


Figure 4.7: Original sphere

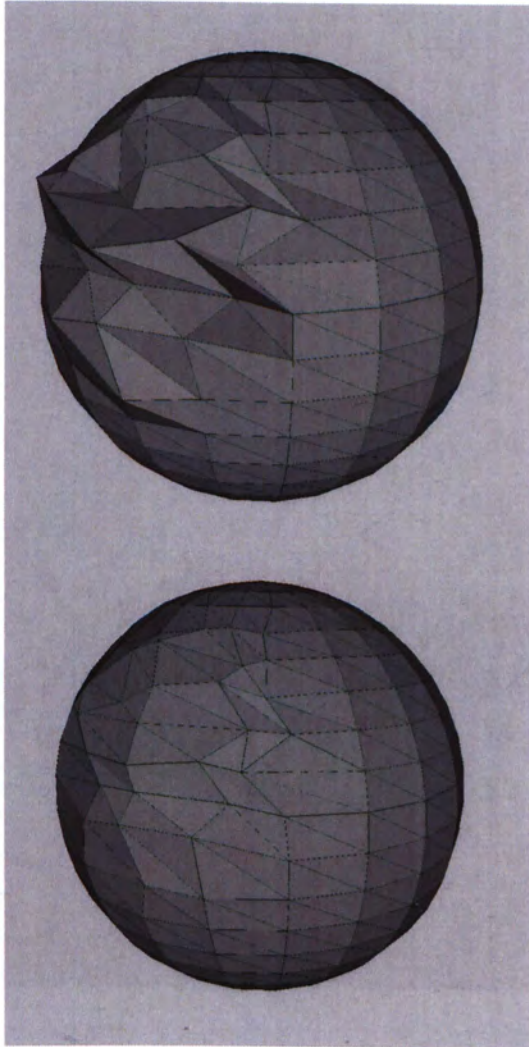


Figure 4.8: Distortion on the sphere (top). Sphere after 60 iterations of the mean-curvature method is applied, where $\lambda = 10$ (bottom).

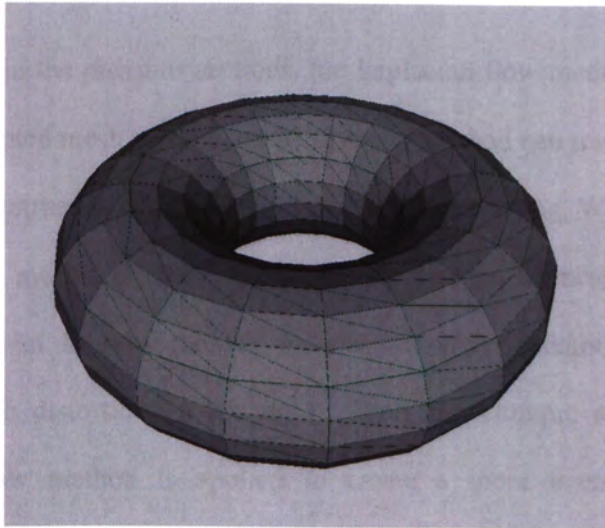


Figure 4.9: Original torus

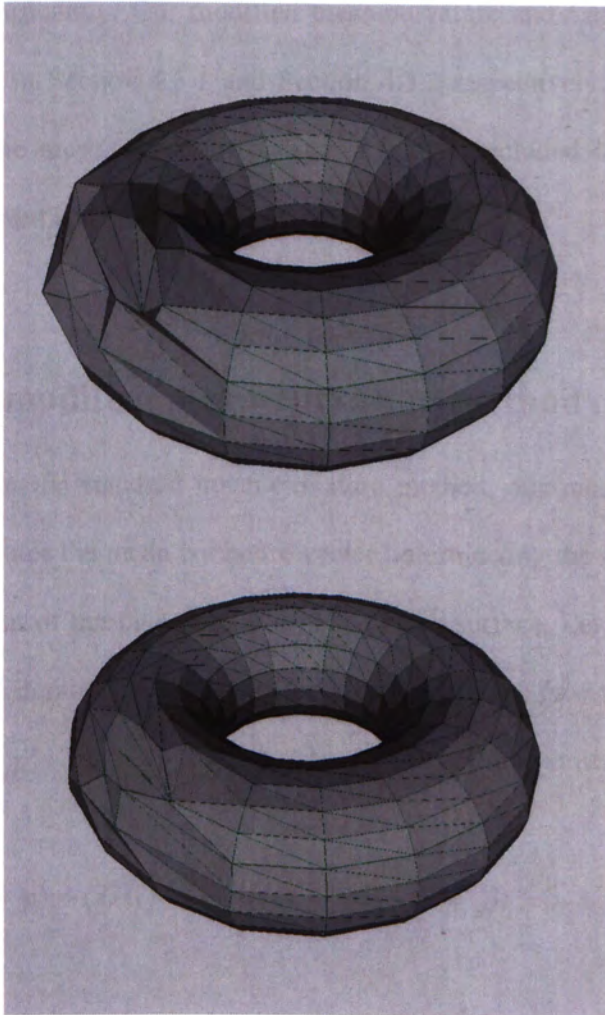


Figure 4.10: Distortion on the torus (top). Torus after 60 iterations of the mean-curvature method is applied, where $\lambda = 10$ (bottom).

4.3 Our Approach

As discussed in the previous sections, the Laplacian flow method can create a more evenly distributed mesh and the mean curvature method can remove the distortion on the mesh. Nevertheless, the object is shrunk after smoothing. We propose a technique to smooth the mesh and create a more evenly distributed mesh without altering the size of the object. In our technique, the mean-curvature method is applied iteratively to remove the distortion on the mesh. Then, a technique similar to that of the Laplacian flow method is applied to create a more evenly distributed mesh. Comparing with the Laplacian flow method, our technique reduces the shrinkage of the mesh significantly. Our modified mean-curvature and Laplacian flow methods are presented in Section 4.3.1 and Section 4.3.2 respectively. In order to preserve features on the mesh, a feature constraint is also included in our technique. The feature constraint is discussed in Section 4.3.3.

4.3.1 The modified mean-curvature method

Different from the standard mean-curvature method, our modified mean-curvature method computes the mean curvature vector by projecting the mean curvature vector in the direction of the face normal \mathbf{n} of the mesh surface. Let p^k be a vertex on the mesh after k^{th} iteration. \mathbf{n} is determined by selecting the face which has the shortest distance with p^k . The new position of the vertex in the next iteration, p^{k+1} , is defined by:

$$p^{k+1} = p^k + (\lambda H(p^k) - (\lambda H(p^k) * \mathbf{n})(\lambda H(p^k))) \quad (4.9)$$

where $\lambda H(p^k) - (\lambda H(p^k) * \mathbf{n})(\lambda H(p^k))$ is the projection of the mean-curvature vector onto the plane of the face defined by p^k and \mathbf{n} , $\{*\}$ is the dot product between two vectors, and p^{k+1} is the intersection between \mathbf{n} and the face defined by p^k on the template mesh (Figure 4.11). The new mesh surface defined by p^{k+1} (the linear segments (red color) in Figure 4.11) is used as a reference surface for the projection in the next iteration.

Comparing with the standard mean-curvature method, our approach reduces the shrinking effect of the object and removes the distortion on the mesh. However, the distribution of the mesh is not improved. As a result, the Laplacian flow method is adopted to create a more evenly distributed mesh.

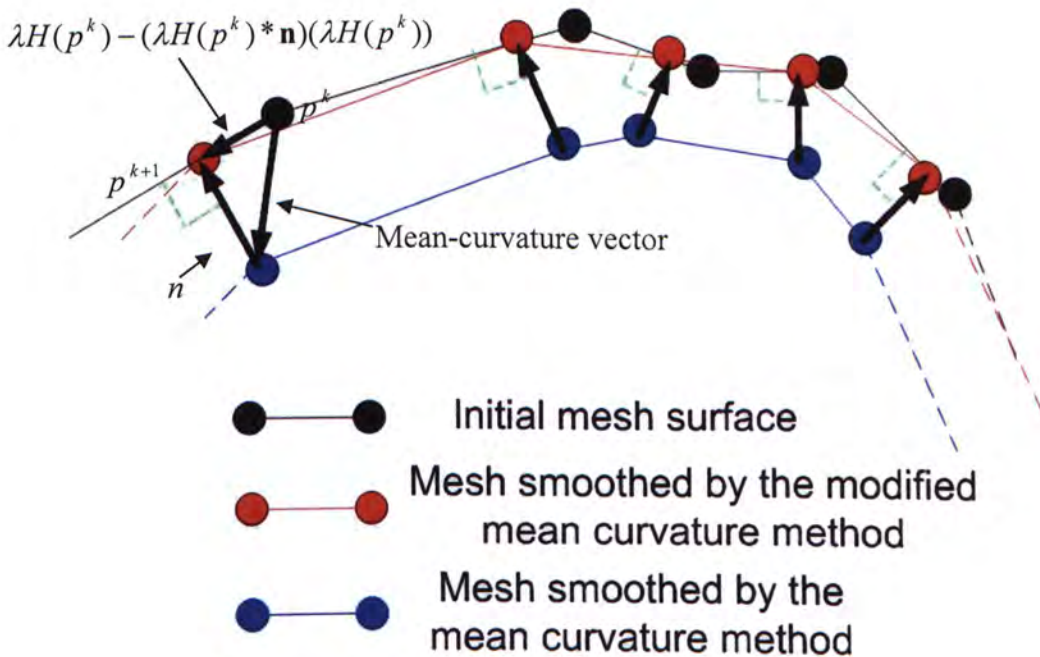


Figure 4.11: Our mean-curvature method

4.3.2 The modified Laplacian flow method

In order to produce a more evenly distributed mesh, an approach similar to that of the Laplacian flow method is proposed. Different from the standard Laplacian flow method, our Laplacian flow method computes the displacement vector $\Delta(p^k)$ (Section 4.1) by projecting $\Delta(p^k)$ in the direction of the face normal \mathbf{n} of the original mesh surface. Let p^k be a vertex on the mesh after k^{th} iteration. \mathbf{n} is determined by selecting the face which has the shortest distance with p^k and $\Delta(p^k)$ is determined by the following equation:

$$\Delta(p^k) = \frac{1}{m} \sum_{i=1}^m q_i - p^k \quad (4.10)$$

where q_i is the i^{th} neighboring vertex of p^k and m is the total number of neighboring vertices of p^k .

The position of the vertex in next iteration, p^{k+1} , is defined by:

$$p^{k+1} = p^0 + (\Delta(p^k) - (\Delta(p^k) * \mathbf{n})(\Delta(p^k))) \quad (4.11)$$

where $\Delta(p^k) - (\Delta(p^k) * \mathbf{n})(\Delta(p^k))$ is the projection of $\Delta(p^k)$ onto the plane of the face defined by the vertex p^0 and \mathbf{n} , and p^{k+1} is the intersection between \mathbf{n} and the face defined by p^0 on the template mesh (Figure 4.12). The original mesh surface defined by p^0 (the linear segments (black color) in Figure 4.12) is used as a reference surface for the projection in each iteration.

Since p^k is moved along the tangent plane of the original faces, it affects the distribution of the mesh rather than the shape of the mesh. Although the modified Laplacian flow method reduces the smoothing effect of the traditional Laplacian algorithm, a more evenly distributed mesh can be created. As the vertex is moved along the tangent plane of the original faces, this largely reduces the shrinking effect.

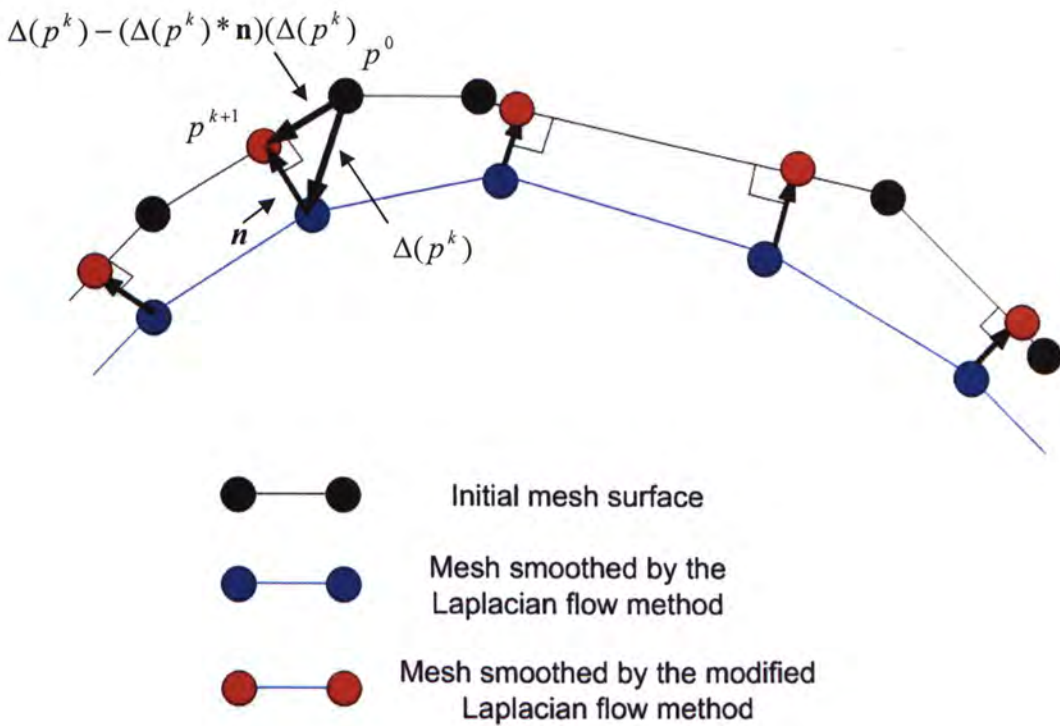


Figure 4.12: Our Laplacian flow method

4.3.3 Feature constraints

In order to keep the features of the original template mesh, some feature constraints are included in the calculation. In our approach, the vertices of the curves which are used as vertices of corresponding pairs with the mesh are defined as the feature points. Let the vertex V_f be the set of feature points, Equations 4.9 and 4.11 now becomes:

For the modified mean curvature method

$$p^{k+1} = \begin{cases} p^k + (\lambda H(p^k) - (\lambda H(p^k) * \mathbf{n})(\lambda H(p^k))) & \text{if } p \notin V_f \\ p^0 & \text{if } p \in V_f \end{cases} \quad (4.12)$$

For the modified Laplacian flow method

$$p^{k+1} = \begin{cases} p^0 + (\Delta(p^k) - (\Delta(p^k) * \mathbf{n})(\Delta(p^k))) & \text{if } p \notin V_f \\ p^0 & \text{if } p \in V_f \end{cases} \quad (4.13)$$

4.3.4 Computational complexity

Let the total number of faces of the template mesh be m_f , the number of iterations involved in our mean-curvature and Laplacian flow method be k_1 and k_2 respectively. For our mean-curvature method, the computational complexity is $O(k_1 m_f)$. For our Laplacian flow method, the computational complexity is $O(k_2 m_f)$. Therefore, the total computational complexity for our smoothing method would be $O(m_f (k_1 + k_2))$.

4.4 Comparison of the mesh smoothing approach

A comparison between the standard Laplacian flow method, mean-curvature method and our approach is presented. In Figure 4.13, the shape of the sphere is changed after using the Laplacian flow method. Since the position of the vertex on the mesh is determined by averaging the position of its neighbors, the distribution of the mesh is improved by using the Laplacian flow method. For the sphere smoothed by the mean-curvature method, the shrinking effect is largely reduced. However, the distribution of triangular faces on the sphere is not improved by using the mean-curvature method. In our technique, the modified mean-curvature method is applied to remove the distortion on the mesh first. Then, the modified Laplacian flow method is applied to create a more evenly distributed mesh. Figure 4.13 shows the result of using our approach. Comparing with the standard Laplacian flow method, the shrinkage of the object is significantly reduced by using our approach. Meanwhile, comparing with the standard mean-curvature method, our technique produces a more evenly distributed mesh. Figure 4.14 shows a similar result of using different approaches on the smoothing of a torus.

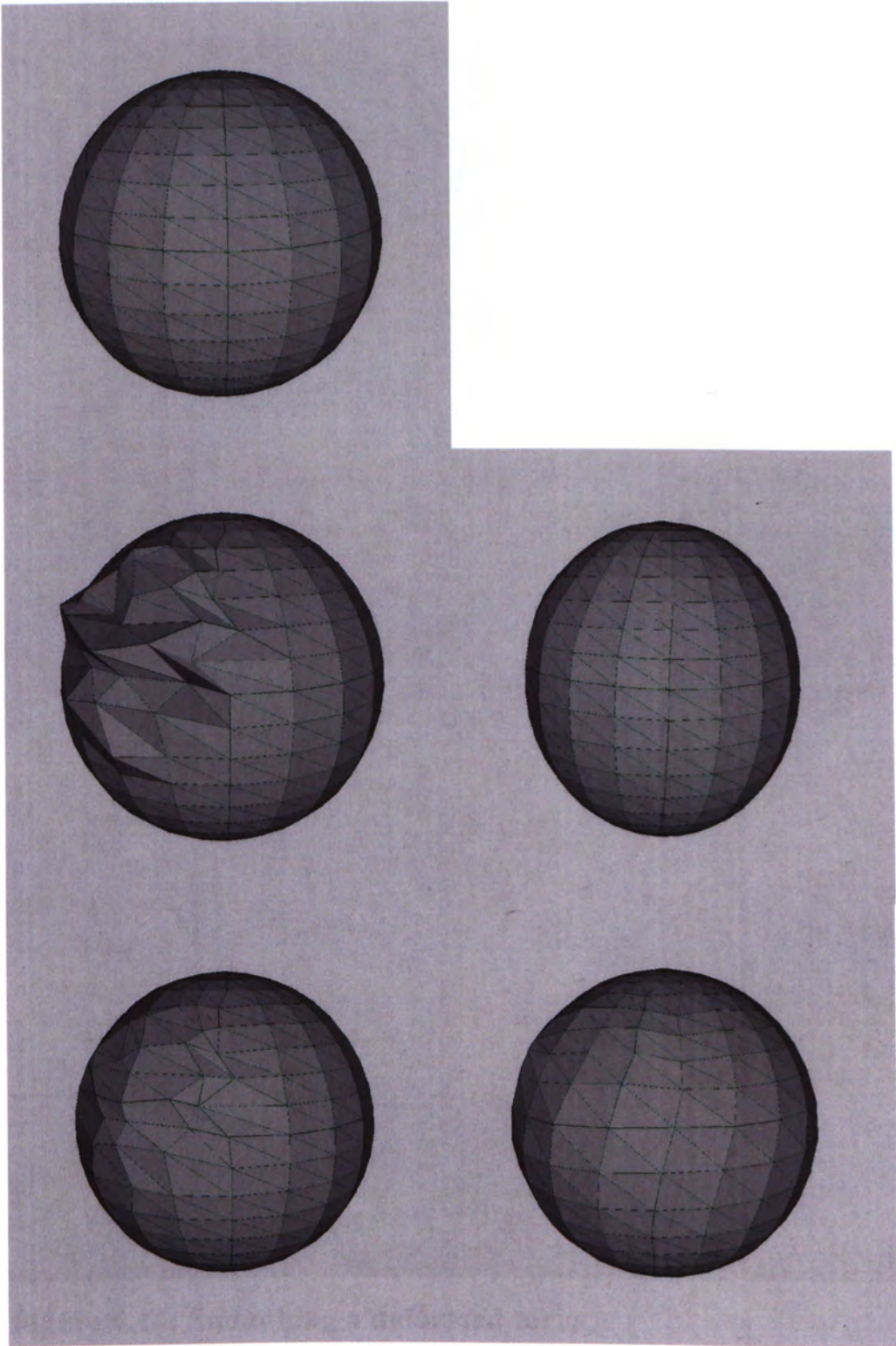


Figure 4.13: Smoothing a deformed sphere. Original sphere (top). Distortions on the sphere (middle left). Sphere after 20 iterations of the Laplacian flow method (middle right), 60 iterations of the mean-curvature method (bottom left) and 100 iterations of our approach (bottom right) is applied

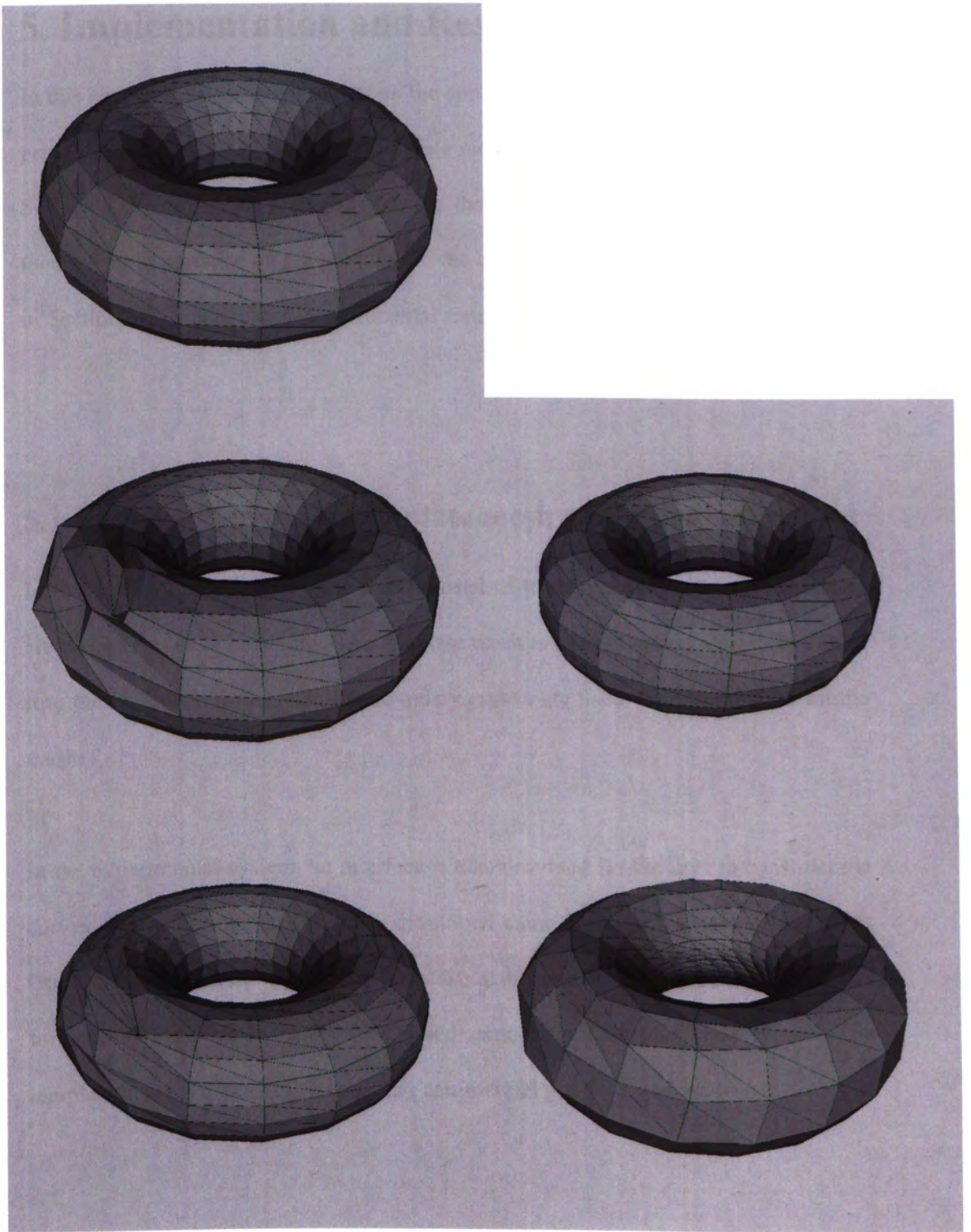


Figure 4.14: Smoothing a deformed torus

Original Torus (top). Distortions on the torus (middle left). Torus after 20 iterations of the Laplacian flow method (middle right), 60 iterations of the mean-curvature method (bottom left) and 100 iterations of our approach (bottom right) is applied

5. Implementation and Results

In this chapter, experimental results on the applications of the proposed technique are presented. In each experiment, the template mesh and the set of curves are specified. Section 5.1 presents the methods to create the template mesh and the curves in this thesis. The importance on the selection of the corresponding vertex pairs is discussed in Section 5.2. Finally, some experimental results are given in Section 5.3.

5.1 Construction of the template mesh and boundary curves

In this thesis, the template mesh is composed of triangular element, and the curves are piecewise linear segments. The template mesh is used to create a mesh by fitting it to the boundary curves, and the boundary curves are used to define regions on the mesh.

In the experimental system, an interface is also provided for the user to construct the curves. The experimental system is developed using Microsoft Visual C++ and the Open Graphic Library (OpenGL). It accepts the template mesh with various file formats such as “ASE” and “OBJ”, and exports models in “OBJ” format. The template mesh is created by an existing commercial graphics system, Maya.

5.2 Selection of the corresponding vertex pairs

In the global deformation, correspondences are determined automatically by the closest point method. However, a set of corresponding vertex pairs are required to be defined manually first. The accuracy of the correspondences determined by the closest point method relies on the accuracy of the manually defined corresponding vertex pairs. In Figure 5.1, the labels marked on the template mesh and the curves represent the manually defined corresponding vertices pairs. Figure 5.1b shows a set of incorrectly selected corresponding vertex pairs between the template mesh and the curves. Figure 5.2 shows the defect if the corresponding vertex pairs are selected incorrectly.

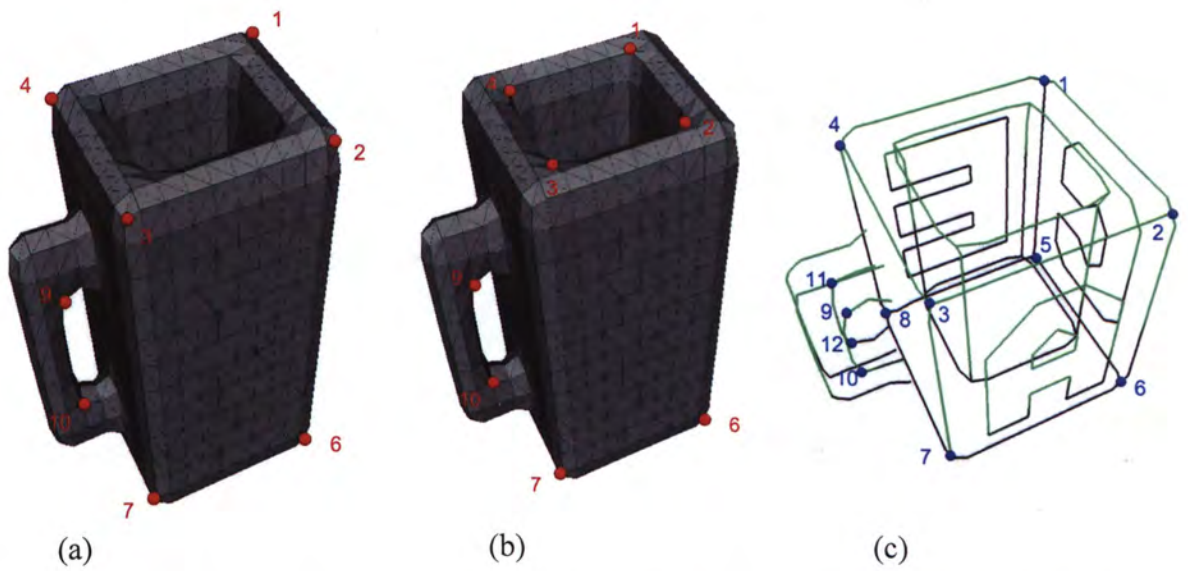


Figure 5.1: The template mesh with a set of corresponding vertices (left). The template mesh with incorrectly selected corresponding vertices (middle). The curves with the corresponding vertices (right).

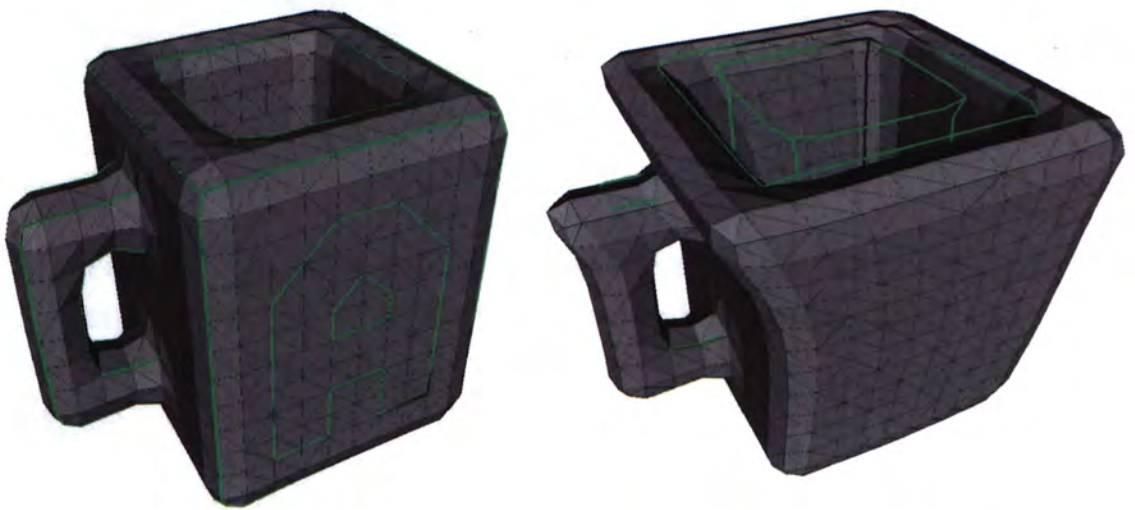


Figure 5.2: The deformed template mesh with a proper set of corresponding vertex pairs (left), and the deformed template mesh with improper corresponding vertex pairs (right)

5.3 Results

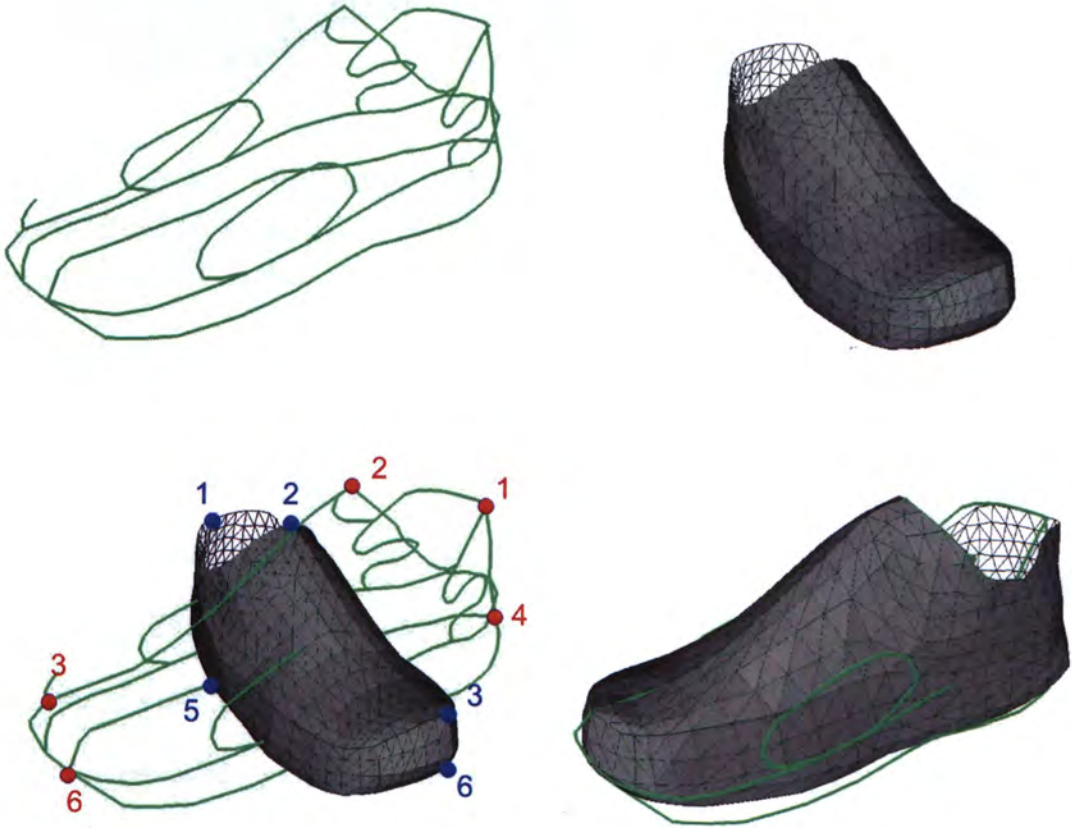


Figure 5.3: The shoe template and the curves network (top). 7 corresponding point pairs are specified (bottom left). The shoe template after the coarse-level global deformation is applied (bottom right).

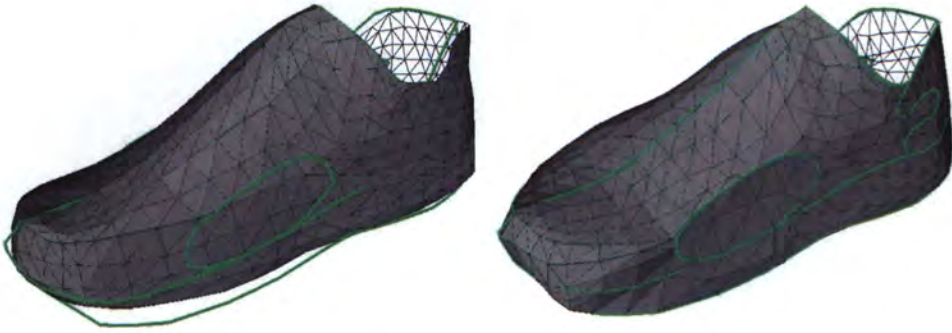


Figure 5.4: The shoe template before the fine-level global deformation is applied (left). The shoe template after the fine-level global deformation is applied (right).

Figure 5.3 shows a shoe template after the coarse-level global deformation. Based on the specified corresponding vertex pairs, the orientation and the shape of the template mesh are adjusted. Figure 5.4 shows the template mesh being fitted to the curves after the fine-level global deformation. Figure 5.5 shows the template mesh partitioned into regions by the curves. As illustrated in Figure 5.6, distortion of the mesh is removed by using the modified mean curvature method. Figure 5.7 shows that the mesh is more evenly distributed after applying the modified smoothing techniques. The final result of the mesh with partitions is shown in Figure 5.8.

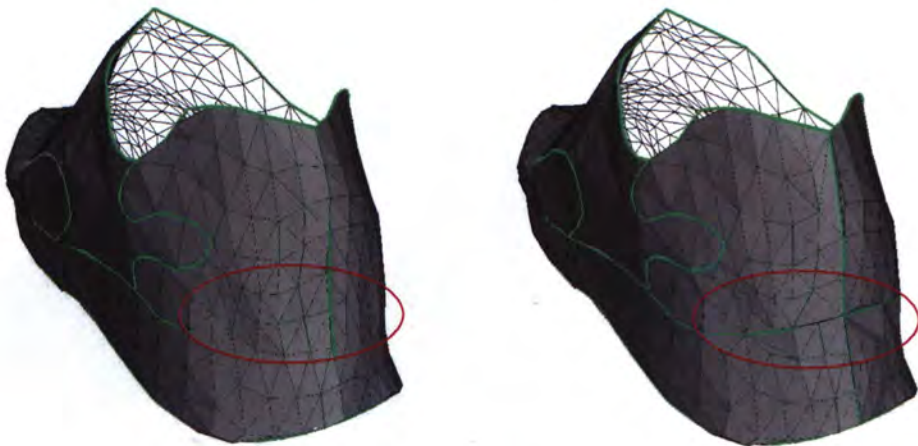


Figure 5.5: The shoe template before (top) and after (bottom) partitioning.

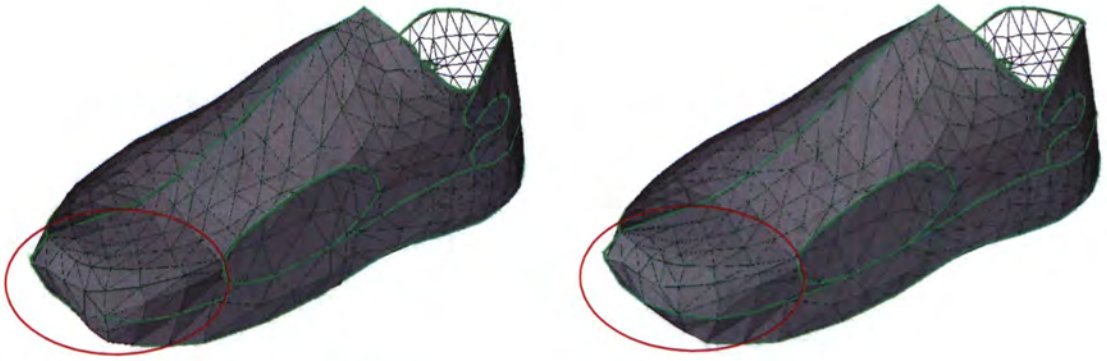


Figure 5.6: The shoe template before smoothing (left). The shoe template after applying the modified mean-curvature method (right).

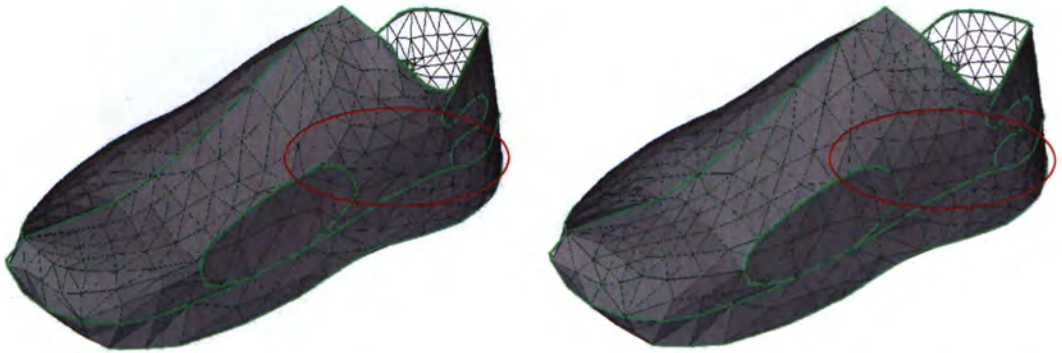


Figure 5.7: The shoe template before smoothing (left). The shoe template after applying the modified Laplacian flow method (right).



Figure 5.8: Final mesh with partitions

Figure 5.9 shows an example of constructing the same cup using a different template mesh. The results demonstrated that the proposed method can create the required mesh by using different template meshes. Figure 5.10 shows the result with fewer feature constraints (Section 4.3.3).

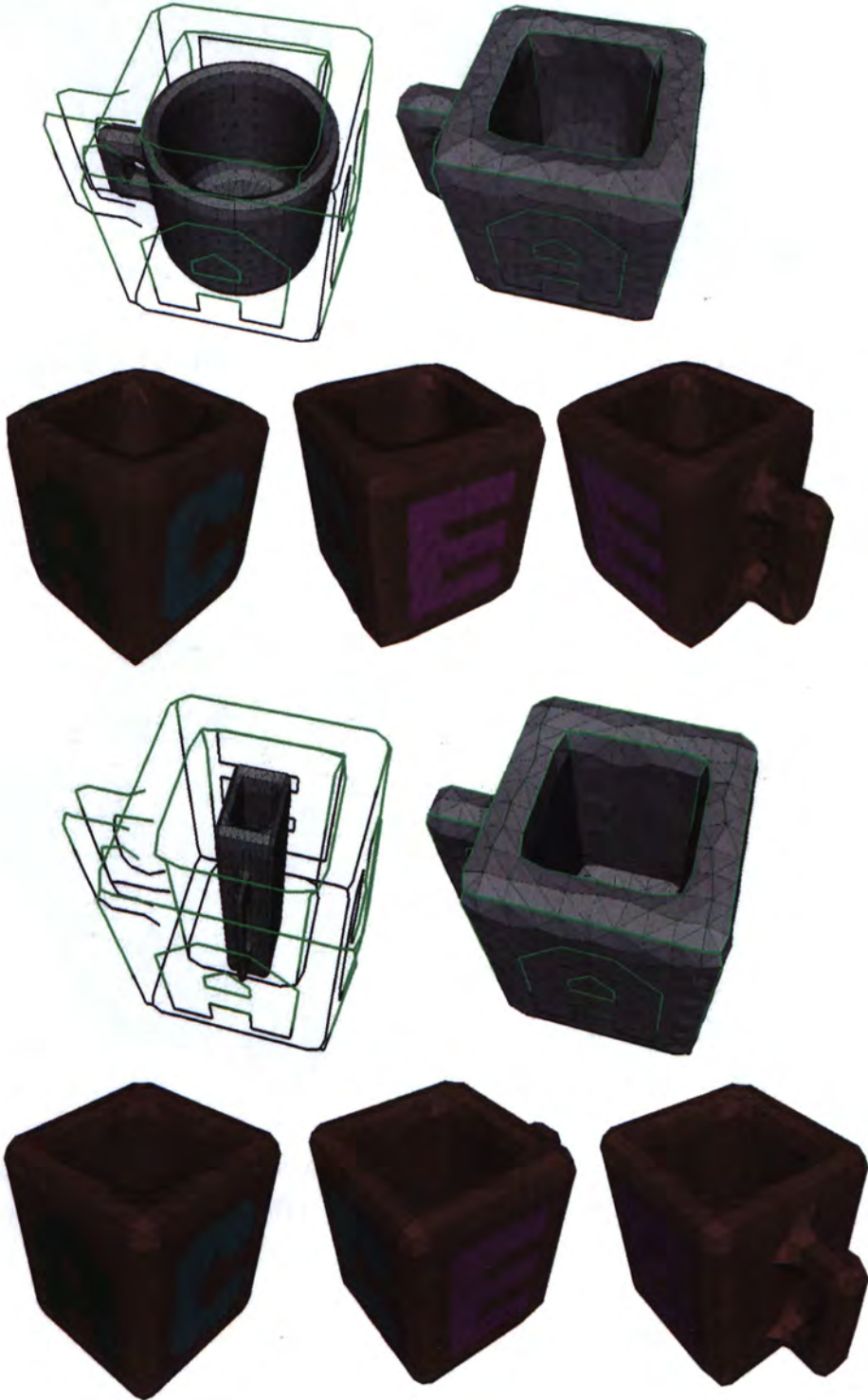


Figure 5.9: Results of using different template mesh

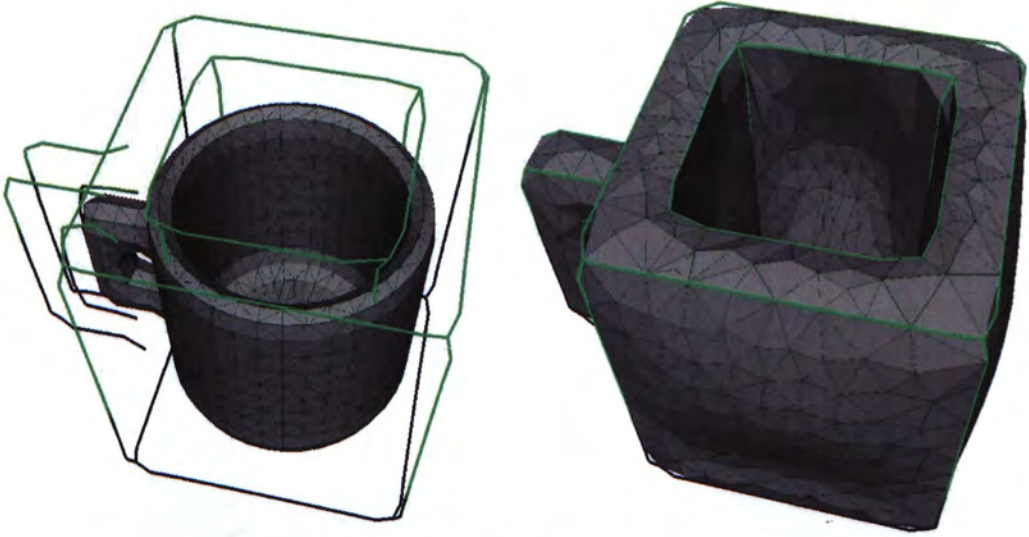


Figure 5.10: Results of using fewer feature constraints

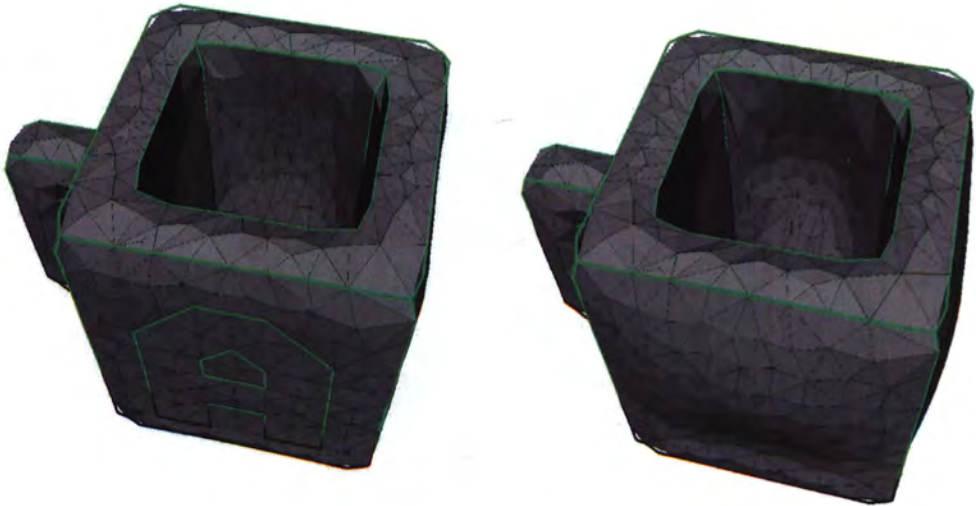


Figure 5.11: The effect of feature constraints. Result with more feature constraints (left). Result with fewer feature constraints (right).

Figure 5.12 shows the result of fitting the template mesh to a set of the curves with concave and convex regions. The result is satisfactory as the basic characteristics of the elephant are captured by the template mesh successfully. Our method is also applied to a template mesh with open surface as shown in Figure 5.13. Figure 5.14 shows the results of fitting different resolutions of the template mesh to the curves.

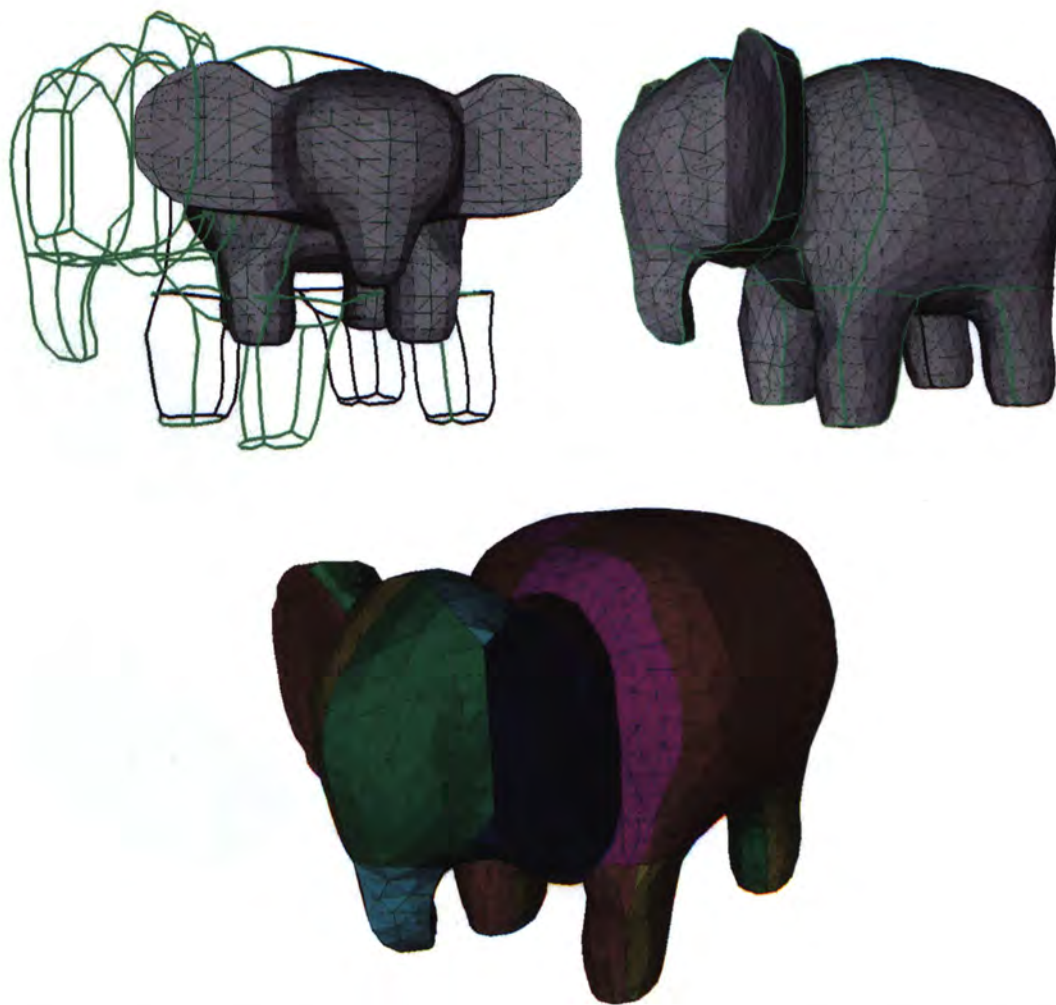


Figure 5.12: The elephant template before deformation (top left). The elephant template after deformation (top right). Final mesh (bottom)

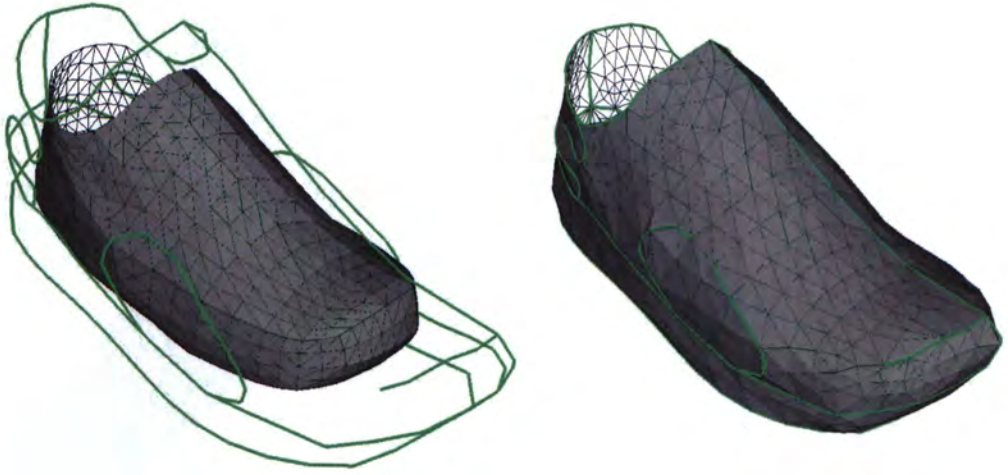


Figure 5.13: Using an open template mesh to fit a set of curves.

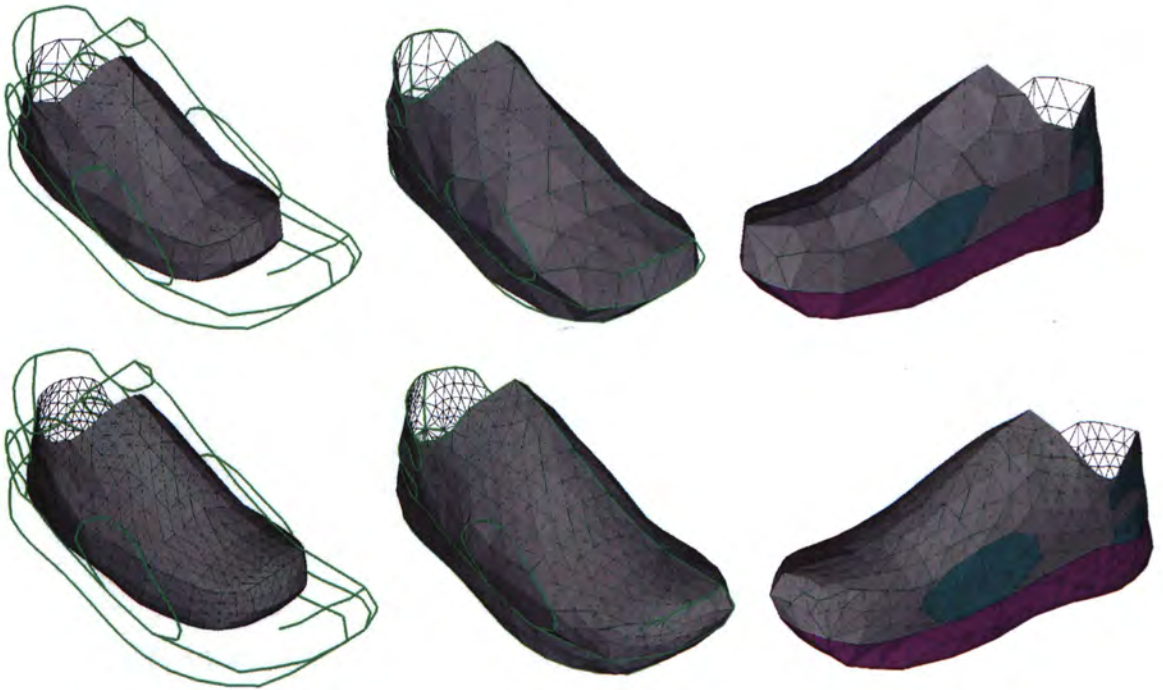
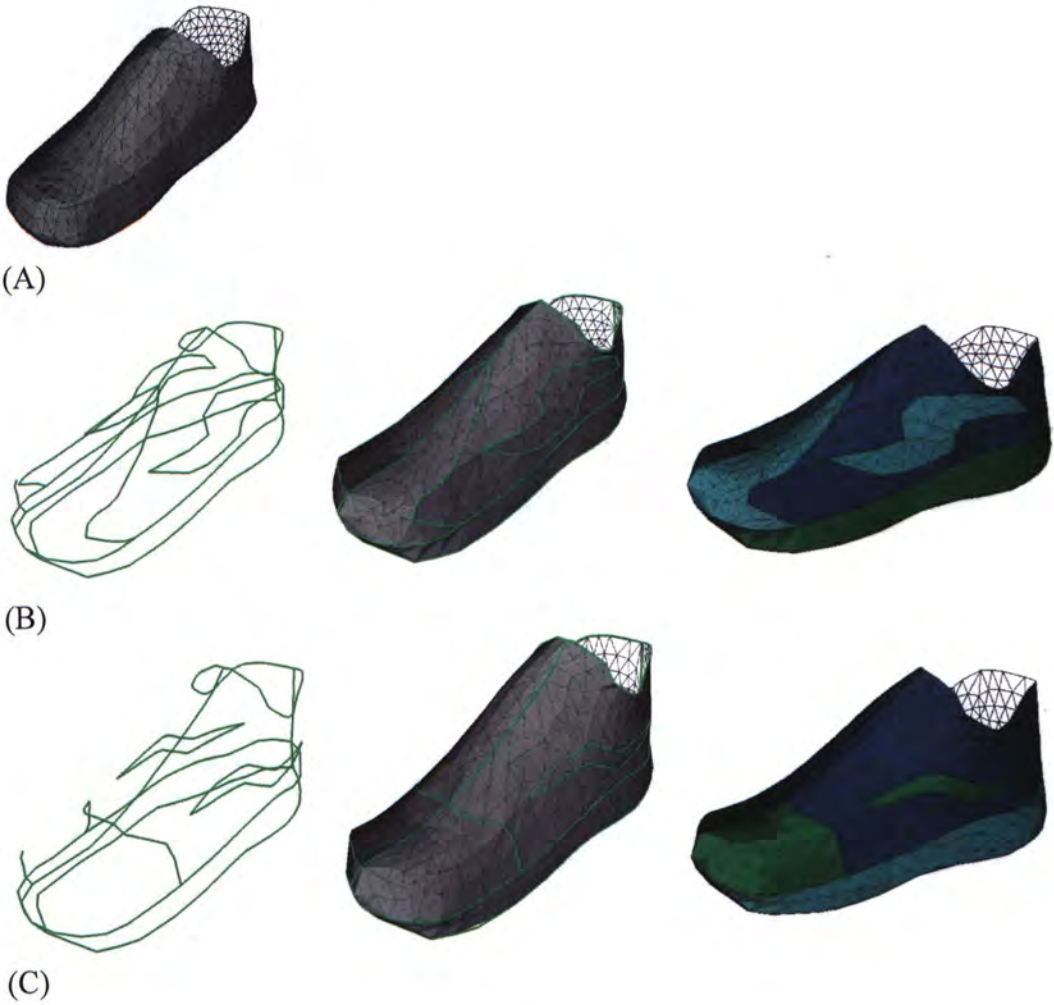


Figure 5.14: Results of using template mesh with different resolutions. The template mesh before deformation (left). The template mesh after deformation (middle). Final mesh (right).

Figure 5.15 shows an example of constructing different shoe models from the same template mesh. The results demonstrated that the proposed method can be used to create different mesh by using the same template mesh. Figure 5.16 shows similar result of fitting the same template mesh to different set of curves.



(A): Initial template mesh. (B) & (C): Different set of curves

Figure 5.15: Fitting a template mesh to different set of curves (1).



(A)



(B)



(C)



(A): Initial template mesh. (B) & (C): Different set of curves

Figure 5.16: Fitting a template mesh to different set of curves (2)

6. Conclusions

In this thesis, a technique to create a mesh with partitions by fitting a template mesh to a set of curves is proposed. The technique is particularly useful in computer graphics and computer-aided design applications. The proposed method consists of three main stages 1) global deformation, 2) mesh partitioning and 3) mesh smoothing. In the first stage, global deformation technique is employed. The global deformation technique includes a coarse-level and a fine-level deformation. Based on a set of user specified corresponding vertex pairs, the orientation and the shape of the template mesh are adjusted in the coarse-level deformation. Then, another set of correspondences between the template mesh and the curves are determined by the closest point method, and the fine-level global deformation is applied to fit the template mesh to the curves. However, some faces are flipped in the deformation. A face flip prevention technique is then presented. In the second stage, the template mesh is partitioned by re-triangulating the mesh in the vicinity of the curves. The advantage of using the re-triangulation process to partition the mesh is that the mesh size is not varied after partitioning. The main drawback of the technique is that the topology of the mesh is changed after the re-triangulation process. Thus induces some distortion on the template mesh. In the final stage, a mesh smoothing technique is adopted. Our smoothing technique is a modified mean-curvature and the Laplacian flow method. The modified mean-curvature method is used to remove the distortion of the mesh and the modified Laplacian flow method is applied to create a more evenly distributed mesh. Experimental results demonstrated that the proposed method generates meshes with partitions while capturing the basic characteristics of the curves successfully.

Despite our proposed method can generate meshes with partitions successfully, specific features of the model may not be retained if the feature is not specified as corresponding vertex pairs. Some further work can be conducted to improve the proposed method. This will be discussed in ~~the~~ Section 6.1.

6.1 Future development

In this research, there are several areas that we can do in the future. They are described as follows:

1. Improvement of the corresponding vertex detection

In order to fit a template mesh to a set of curves, corresponding vertex pairs between the template mesh and the curves are determined. For a template mesh constructed with open surface, vertices lying on the boundary of the mesh must be included in the set of corresponding vertices. However, in our approach, not all vertices on the boundary of the mesh can be determined as corresponding vertices by using the closest point method. It is expected that a feature recognition technique can be applied to detect feature vertices of the template mesh [33]. The closest point method can then be applied to establish correspondence between the feature vertices of the mesh and the vertices of the curves. Since the closest point method is only applied to the feature vertices of the mesh, correspondence will only be established for feature vertices and hence avoids matching boundary curve points to vertices not lying on the boundary of the mesh.

2. Preservation of feature on the template mesh

In this research, the feature on the template mesh will be removed after smoothing if the corresponding feature vertex is not specified on the curves. It is expected the feature recognition technique [33] can be applied to detect the feature of the template mesh. The recognized feature vertex can then be used as feature constraints (Section 4.3.3) during smoothing. Thus, the feature of the template mesh can be preserved after smoothing.

Appendix A

Determination of the projected path on a mesh:

In this section, the technique to determine the projected path on a mesh is proposed. Our approach accepts a linear segment and a template mesh as inputs. The goal is to determine the projected path of the linear segment on the mesh. After the RBFs deformation (Chapter 2), the vertices on the linear segment are vertices on the mesh. In this case, the projected path can be determined if the direction of the linear segment is specified.

Let the face normal, the direction of the linear segment and the direction of the projected path on the mesh be \mathbf{n} , dir_c and dir_p respectively (see Figure A.1). By determining dir_c using:

$$dir_c = v_s - v_e \quad (A.1)$$

where v_s and v_e are the starting point and the end point of the linear segment respectively. dir_p can be determined by:

$$dir_p = \mathbf{n} \times (dir_c \times \mathbf{n}) \quad (A.2)$$

The steps of determining the projected path on the mesh are summarized:

1. determine the face on the mesh which contains the starting point of the linear segment, and check for any intersection between dir_p and the boundaries of the face.
2. if there is intersection between dir_p and the boundaries of the face, the face is defined as the starting face. We define the starting point of the linear segment

- as the starting point of the projected path and the intersection point I as the second vertex on the projected path.
3. search the neighboring face of the starting face by checking the intersecting edge.
 4. recalculate dir_c by setting v_s equal to I in Equation (A.1)
 5. update n with the normal of the neighboring face in Step3. The neighboring face is then defined as starting face.
 6. recalculate dir_p by using Equation (A.2).
 7. update I as the intersection point between dir_p and the boundaries of the starting face, and I is then defined as the vertex on the projected path.
 8. iterate Step 3-7 until I is equal to v_e .

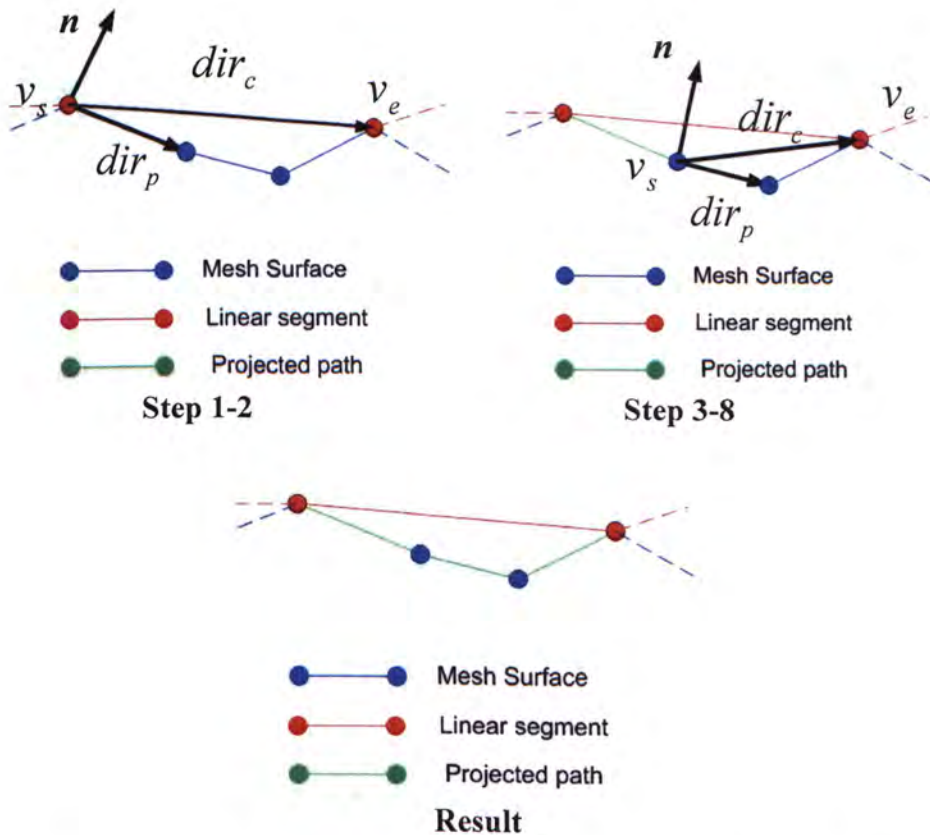


Figure A.1: Determination of the projected path on the mesh.

Reference

- [1] J. C. Cavendish, D. A. Field and W. H. Frey, "An approach to automatic three-dimensional finite element mesh generation", *Int. j. numer. methods eng.*, 21, 329-347 (1985).

- [2] M. M. F. Yuen, S. T. Tan and K. Y. Hung, "A hierarchical approach to automatic finite element mesh generation", *Int. j. numer. methods eng.*, 32, 501-525 (1991).

- [3] W. J. Schroeder and M. S. Shephard, "Geometry-based fully automatic mesh generation and the Delaunay triangulation", *Int. j. numer. methods eng.*, 26, 2503-2515 (1988)

- [4] S. H. Lo, "Volume discretization into tetrahedra - II. 3D triangulation by advancing front approach", *Comput. & Struct.*, 39, 501-511 (1991).

- [5] J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O. C. Zienkiewicz, "Finite element Euler computations in three dimensions", *Int. j. numer. methods eng.*, 26, 2135-2159 (1988).

- [6] P. Vijayan and Y. Kallinderis, "A 3D finite-volume scheme for the Euler equations on adaptive tetrahedral grids", *J. Comput. Phys.*, 113, 249-267 (1994).

- [7] Boris, N. Delaunay, (1934) "Sur la Sphere" *Vide. Izvestia Akademia Nauk SSSR*, VII Seria, Otdelenie Matematicheskii I stestvennyka Nauk Vol 7 pp.793-800
- [8] Kolja Kähler, Jörg Haber, Hitoshi Yamauchi, Hans-Peter Seidel. Head shop: Generating animated head models with anatomical structure. Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 55 – 63.
- [9] Kolja Kähler, Jörg Haber, Hans-Peter Seidel. Reanimating the Dead: Reconstruction of Expressive Faces from Skull Data. *ACM Transactions on Graphics*, Volume 22, Issue 3, July, 2003, pp. 554 – 561.
- [10] D. L. Page, A. Koschan, Y. Sun, J. Paik, and M. A. Abidi. Robust Crease Detection and Curvature Estimation of Piecewise Smooth Surfaces from Triangle Mesh Approximations Using Normal Voting. Proceedings of the International Conference on Computer Vision and Pattern Recognition, volume 1 (December 2001), pp. 162 – 167.
- [11] KAHLER, K., HABER, J., YAMAUCHI, H., AND SEIDEL, H.-P. 2002. Head shop: Generating animated head models with anatomical structure. In Proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation, ACM SIGGRAPH, San Antonio, USA, S. N. Spencer, Ed., Association of Computing Machinery (ACM), 55-64.
- [12] KANAI, T., AND SUZUKI, H. 2001. Approximate shortest path on a polyhedral surface and its applications. *Computer Aided Design* 33, 11 (September), 801-811.

- [13] J. Mitchell, D. Mount, and C. Paradimitiou. The discrete geodesic problem. In *SIAM J. Computing*, 1987.
- [14] S. Kapoor. Efficient computation of geodesic shortest paths. In *32nd Annual ACM Symposium on Theory of Computing*, 1999.
- [15] J.Chen and Y.Han. Shortest paths on a polyhedron. In *Sixth ACM Symposium on Computational Geometry*, 1990
- [16] DeRose, T., Duchamp, T., Hoppe, H., McDonald, J. and Stuetzle, W. (1993), Mesh optimization, *Proceedings of SIGGRAPH*, 19-26
- [17] Gross, M. and Hubeli, A. (2000), Fairing of nonmanifolds for visualization, *Proceedings of IEEE Visualization*, 407-414.
- [18] G.Taubin. A signal processing approach to fair surface design. In *Computer Graphics (SIGGRAPH 95 Proceedings)*, pages 351-358. 1985
- [19] M. Desbrum, M. Meyer, P. Schroder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Computer Graphics (SIGGRAPH 99 Proceedings)*, pages 317-324, 1999.
- [20] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge Univ. Press, 1999.

- [21] Ohtake Y, Belyaev A, Bogavski I. Mesh regularization and adaptive smoothing. *Computer-Aided Design*, 2001, 33: 789-800.
- [22] J. Vollmer, R. Mencl, H. Muller, Improved Laplacian Smoothing of Noisy Surface Meshes, Research Report No. 711/1999, June 1999.
- [23] Yutaka Ohtaka, Alexander G. Belyaev, Ilia A. Bogaevski, Polyhedral Surface Smoothing with Simultaneous Mesh Regularization, *Geometric Modeling and Processing 2000*, pp. 229-237.
- [24] Brett Allen, Brian Curless and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transaction On Graphics*, volume 22, issue 3 (July 2003), pp. 587 – 594
- [25] Pighin F., Hecker J., Lischinski D., Szeliski R., and Salesin, D.H. Synthesizing Realistic Facial Expressions from Photographs. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pp. 75-84.
- [26] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright and B.C. McCallum Reconstruction and Representation of 3D Objects with Radial Basis. *Proceeding of the 28th annual conference on Computer Graphics and Interactive Technique*, 2001, pp 67-76.
- [27] R.Bellman. *Introduction to Matrix Analysis*. McGraw-Hill, 1960

- [28] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, Numerical Recipes in C++, The Art of Science Computing, Second Edition.
- [29] L. Kobbelt, S. Campagna, J. Vorsatz, and H. -P. Seidel. Interactive multiresolution modeling on arbitrary meshes. In Computer Graphics (SIGGRAPH 98 Proceedings), pages 105-144, 1998.
- [30] Taksahi Kanai, RIKEN. Parametric Curves on Meshes. Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia, 2005, pp 413-416.
- [31] Wan-Chiu Li, Bruno Levy, Jean-Claude. Mesh Editing with an embedded network of curves. Proceedings of the International Conference on Shape Modeling and Applications 2005 (SMI' 05) - Volume 00, pp 62 -71.
- [32] FELDMAR, J., AND AYACHE, N. 1994. Rigid and affine registration of smooth surfaces using differential properties. In ECCV(2), 397-406.
- [33] Xiangmin Jiao and Michael T. Heath, Feature Detection for Surface Meshes, Computational Science and Engineering, University of Illinois at Urban-Champaign, Urbana, IL 61801, USA.

CUHK Libraries



004366673