國立交通大學

多媒體工程研究所

碩 士 論 文

根基於階層化架構以及區域最佳化的即時細節保持變形模擬

Interactive Detail-preserving Deformation Based on Hierarchical Structure and Region Optimization

研　究　生：張智翔

指導教授：莊榮宏　教授

　　　　　林文杰　教授

中 華 民 國　九 十 七　年 十 月

根基於階層化架構以及區域最佳化的即時細節保持變形模擬
Interactive Detail-preserving Deformation Based on Hierarchical Structure and Region Optimization

研 究 生：張智翔　　Student：Jyh-Shyang Chang
指導教授：莊榮宏　　Advisor：Jung-Hong Chuang
林文杰　　Advisor：Wen-Chieh Lin

國 立 交 通 大 學
多 媒 體 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Multimedia Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

October 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年十月

根基於階層化架構以及區域最佳化的即時細節保持變形模擬

研究生 ： 張智翔　　　　　　指導教授 ： 莊榮宏 博士

林文杰 博士

國立交通大學

資訊學院多媒體工程研究所

摘 要

我們提出了一個利用簡化網格來模擬原始網格變形的方法，能夠達到即時的細節保持變形模擬，且在即時的要求下不失去對網格細節的操作。我們利用階層化架構來確保對網格細節的操作以及利用區域最佳化來達到即時運算。在我們的系統中，原始網格首先被化簡為簡化網格，在化簡的過程中建構階層化架構以及在原始網格和簡化網格之間頂點和頂點的對應關係。當簡化網格變形時，我們在階層式架構中考量變形的劇烈程度、最佳化區域邊界的狀況、細節保持的品質以及執行效率，找到一個網格的式樣(cut mesh)，此式樣相對於原始網格而言是一個比較簡化的網格。透過階層化架構以及網格式樣選取的考量，我們可以將原始網格切割為數個各自獨立的區域，這些區域會切割出網格上的細節或是變形劇烈的部分，針對每個區域各自做最佳化的動作來呈現原始網格的變形。

Interactive Detail-preserving Deformation Based on Hierarchical Structure and Region Optimization

Student: Jyh-Shyang Chang Advisor: Dr. Jung-Hong Chuang

Dr. Wen-Chieh Lin

Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

ABSTRACT

We propose an interactive coarse mesh driven detail preserving mesh deformation method. We can achieve interactive performance without losing detail control based on hierarchical structure and region optimization. In our system, mesh is first simplified to base mesh, hierarchy and mapping relationship are built in preprocessing stage. During runtime stage, dynamically finding a cut on hierarchy to control trade-off between performance and quality. Cut is designed to catch acute deformation, maintaining region boundary smoothness and preserving detail on mesh. Mesh is subdivided into regions according to cut. Details and simulation features are captured by regions and these regions are reconstructed though an optimization scheme. We achieve interactive performance by region optimization control and resolution of deformation is not limited because of hierarchical information.

Acknowledgments

I would like to thank my advisors, Professor Jung-Hong Chuang, and Wen-Chieh Lin for their guidance, inspirations, and encouragement. I am grateful to Professor Sai-Keung Wong and PH.D senior Tan-Chi Ho for their comments and advices. Thanks to my colleagues in CGGM lab and Professor Lin's lab: Yueh-Tse Chen works hard himself and press me to work harder, Ying-Tsung Li help me on the programming problems, Hsin-Hsiao Lin help me on the integration problems of applying different engines on my work, Kuang-Wei Fu help me to release my pressure during these time, Hong-Xuan Ji and Cheng-Min Liu for his kind support, Yi-Jheng Huang for her model support, Ying Yi Chou for his engine survey, Shang-Wen Wang, Kuei-Li Fang, Ta-En Chen and Jau-An Yang. It is pleasure to be with all of you in the pass two years. Lastly, I would like to thank my parents for their love, and support. Without their support I could not come through all the pain in my life.

Contents

1 Introduction **1**

1.1 Contribution 2

1.2 Outline 3

2 Background **4**

2.1 Finite element method 4

2.2 Mass-spring system 6

2.3 Meshless method 7

2.4 Multi-resolution techniques 8

2.5 Differential coordinate 8

2.5.1 Representation 10

2.5.2 Optimization 11

3 Detail Preserving Deformation **14**

3.1 System Structure 16

3.1.1 Build hierarchy 17

3.1.2 Build mapping relationship 23

3.2 Differential coordinate representation 24

3.3 Cut updating criterion 26

3.3.1 Geometric stretch 27

3.3.2 Normal difference 27
3.3.3 Performance threshold 29
3.4 Directly recover 30
3.5 Region optimization 30

4 Results and Discussion **34**
4.1 Detail preserving deformation results 34
4.1.1 Performance and quality 38
4.2 Discussion 41

5 Conclusion **54**

Bibliography **56**

List of Figures

1.1 Deformation with or without detail-preserving. 2

1.2 Purple line express cut in hierarchy. Directly recover method is applied on nodes above cut(green nodes) and sub-region optimization is applied on nodes below cut which are colored by region. 3

2.1 Coarse control meshes around the original fine meshes [HSL$^+$06]. 10

2.2 Laplacian coordinate for vertex i. 11

2.3 Laplacian operator L. 12

2.4 Example of solving equation 2.5. 13

3.1 System structure. 15

3.2 System structure example. 16

3.3 Our method applied on free-form deformation. 17

3.4 Our method applied on skeleton-based deformation. 18

3.5 Level of detail example on Armadillo model. 19

3.6 Mesh simplification example. Green parts are claimed as details. 19

3.7 A vertex removal operation. 20

3.8 Label condition after vertex removal. 21

3.9 Tree construction. 21

3.10 Vertex removal and building hierarchy. 22

3.11 2D triangulation using MAPS [LSS+98]. 24
3.12 MAPS Parameterization [LSS+98]. 25
3.13 Local coordinate system transformation. 26
3.14 Geometric stretch. 28
3.15 Normal difference. 28
3.16 Vertex recover operation divide one region into two. Green edges are new appeared region boundary. 29
3.17 Performance calculation. 29
3.18 Performance estimation: $T = 3^2 + 3^2 + 3^2 + 1^2 + 1^2 + 7^2 + 7^2$. 30
3.19 Directly recover example. 31
3.20 Region optimization with boundary constraint. 32
3.21 Region optimization example 33

4.1 Armadillo 10k model deformation result. 35
4.2 Armadillo 10k model deformation result. 36
4.3 10k model deformation results. 37
4.4 Feature is not smoothed under smaller stretch. 38
4.5 Comparison of 5k bumpy sphere deformation model using PolyMeCo. 39
4.6 Comparison of 10k Armadillo deformation model using PolyMeCo. 40
4.7 Visualization of experiment on model with 5000 vertices. 45
4.8 Deformation example on 2D. 46
4.9 Comparison of the region optimization time with the same threshold between different models. 47
4.10 Region visualization. 47
4.11 Simulation time Comparison between directly recover(1) or optimization(2) applied on cut mesh. 47
4.12 Comparison between directly recover(left) or optimization(right) applied on cut mesh. 48

4.13 Different cut update criterion. 49
4.14 Optimized region vertex number on different cut updating criterion as shown in Figure 4.13. 50
4.15 Vertex hierarchy and region. 51
4.16 Build vertex hierarchy upon vertex removal. 52
4.17 Vertex disconnection is happened on sub-tree region when using vertex removal operation. 53

List of Tables

4.1 Performance and quality measurement on model with 2500 vertices. 41

4.2 Performance and quality measurement on model with 5000 vertices. 42

4.3 Performance and quality measurement on model with 7500 vertices. 43

4.4 Performance and quality measurement on model with 10000 vertices. 43

4.5 Performance and quality measurement on model with 12500 vertices. 44

4.6 Performance and quality measurement on model with 15000 vertices. 44

4.7 Performance and quality measurement on model with 5000 vertices with different approach on cut mesh. 2000 vertices are directly recovered(or optimized) and 2000 vertices are reconstructed through region optimization. 45

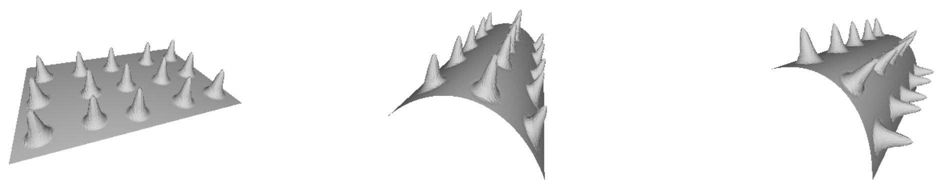
CHAPTER 1

Introduction

We propose an interactive detail-preserving structure based on hierarchical structure and region optimization. Local detail properties are represented by differential coordinates. The term ”detail-preserving” is first explained in [LSC$^+$04]. ”Differential coordinate is an alternative representation for the vertex coordinate. The use of linear differential coordinates as mean to preserve the high-frequency detail of the surface. The shape and orientation of these details are preserved. The differential coordinates represent the details and are defined by a linear transformation of the mesh vertices. This representation allows a direct detail-preserving reconstruction of the modified mesh by solving a linear least squares system.”

The main objective on detail-preserving algorithms is to preserve shape and orientation of geometric details. An example showing difference between general deformation and detail-preserving deformation is shown in Figure 1.1. The deformation result using detail-preserving approach is more visually acceptable.

Our detail-preserving deformation method seeks balance between performance and quality at runtime stage. Our work uses the base mesh deformation to drive the original mesh deformation. In our system, an input original mesh is simplified into a base mesh at pre-processing stage. The criterion we choose for simplifying mesh is that we simplify *details* which are locally



(a) Original mesh. (b) Deformation without detail-preserving. (c) Deformation with detail-preserving.

Figure 1.1: Deformation with or without detail-preserving.

high frequency features determined by curvature and mesh saliency. A hierarchical structure with dependency is constructed during simplification. Every node(except root) on the hierarchical structure represents a vertex removal/recover operator. MAPS parameterization is built at pre-processing stage for rotation assumption of differential coordinate. At dynamic stage, cut updating criterion are based on deformation properties, region smoothness, detail-preserving quality and performance. A cut found on the hierarchical structure represents a cut mesh (A coarser mesh compared to original mesh). Directly recover method is applied on nodes above cut to get cut mesh deformation. Sub-region optimization scheme is applied on nodes below cut to get original mesh deformation as shown in Figure 1.2.

1.1 Contribution

Our structure achieves dynamic interactive performance without losing detail control. With a hierarchical structure, finding a cut on hierarchy to catch global appearance and details are reconstructed through region optimization. Cut finding criterion is designed to seek balance between performance and quality. Deformation resolution is not limited because of hierarchical information.

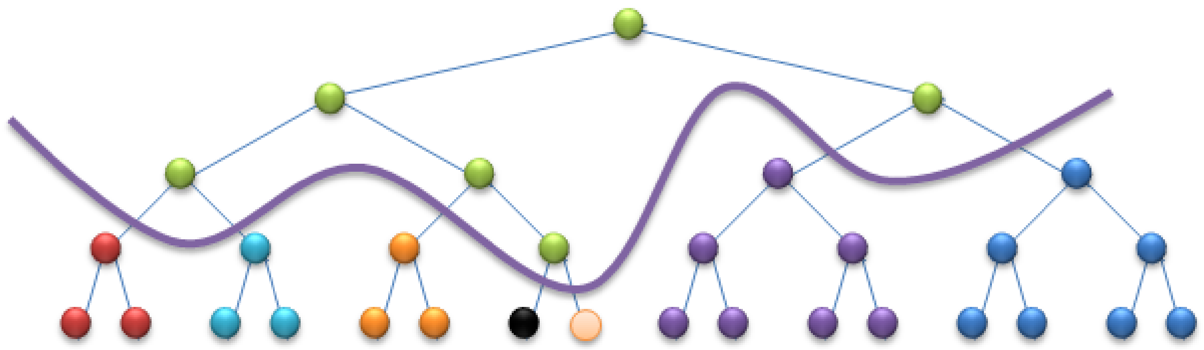


Figure 1.2: Purple line express cut in hierarchy. Directly recover method is applied on nodes above cut(green nodes) and sub-region optimization is applied on nodes below cut which are colored by region.

1.2 Outline

The rest of the thesis is organized as follows : Chapter 2 gives the background on different deformation methods. Chapter 3 presents our detail-preserving deformation method: At pre-processing stage we build hierarchy and mapping relationship. During runtime stage, cut updating criterion, directly recover method and region optimization method. Chapter 4 shows our experimental results from our approach. In the end, conclusions and future work are discussed in Chapter 5.

CHAPTER 2

Background

The first approach of physically-based deformable models in computer graphics is researched by Terzopoulos et al. Their approaches using Lagrangian equations and finite differences method to simulate elastic [TPBF87] and inelastic [TF88] behaviors. However, elements become numerically ill-conditioned when handling large deformation. So further research combining rigid body motion term to increase stability with stiff bodies [TW88]. Finite element method is also been used in various application: modeling a hand grasping a ball [GTT89], simulating muscles [CZ92] and virtual surgery [PDA01].

2.1 Finite element method

The finite element method(FEM) is one of the most popular methods in computational sciences to solve Partial Differential Equations(PDE) on elements. Continuous object is viewed as union of discrete connected elements. With continuum mechanics between elements, behavior on continuous object can be integrated from discrete elements' behavior. The finite element method turns a PDE into a set of algebraic equations which can be solved numerically.

The PDE represents dynamic elastic materials behavior is given by

$$\rho\ddot{x} = \bigtriangledown \cdot \sigma + f \tag{2.1}$$

where ρ is the density of material and f represents externally applied forces. The divergence operator represents internal force resulted from deformed object.

$$\bigtriangledown \cdot \sigma = \begin{bmatrix} \sigma_{xx,x} & \sigma_{xy,y} & \sigma_{xz,z} \\ \sigma_{yx,x} & \sigma_{yy,y} & \sigma_{yz,z} \\ \sigma_{zx,x} & \sigma_{zy,y} & \sigma_{zz,z} \end{bmatrix}$$

Finite element methods are commonly used to compute the elastic solid behavior. O'Brien et al. simulated brittle fracture [OH99], using FEM on calculating element boundary deforming energy to decide where to crack. He also extended these methods with a simple plasticity model [OBH02], the range of simulating materials are thereby increased. They use tetrahedra as discrete elements on mesh with connection by linear basis function and Green's strain tensor. The resulting nonlinear equations are solved explicitly and integrated explicitly. The method produces realistic and visually convincing results, but it is not designed for interactive or real-time use. For example, O'Brien et al. fractures object along elements' boundary [OH99], resolution of volume elements will affect visually appearance. To achieve visually convincing results, object should be divided into large amount of volume elements and results in lots of computation time. Müller et al. achieves interactive performance by choosing cube as volume elements to broadly calculate where to crack and remesh naked parts to achieve visually convincing results [MTG04].

However, a situation that causes simulation to become unstable is that numerical calculation becomes ill-conditioned. It may happens on large deformation or irregular element deformation. Irving et al. [ITA04], built on the work of Müller et al. ([MDM$^+$02] and [MG04]), to create a finite element method that robustly handles inverted elements. Using singular value decomposition(SVD) to separate the rotational and scaling components of the deformation gradient. By constraining the deformation gradient matrix to become well-conditioned, it makes finite element methods more stable for computer graphics applications. But their approach does not address the problems that result from ill-conditioned basis matrices. Consequently, it can handle only limited amounts of plastic flow. Bargteil [BWHT07] maintain well-conditioned basis matrices by remeshing the simulation mesh. The simulation domain is remeshed to produce a new high-quality finite element mesh when finding some ill-conditioned situations, taking care to preserve the original boundary. Remeshing during finite element simulations now first introduced to computer graphics. But it's hard to achieve interactive performance by large dynamic remesh computation. An interesting alternative to finite element was proposed by Clavet et al. [CBP05]. In their approach, an object is treated as a mass spring system in which the spring are dynamically inserted and removed. The process is similar to the remeshing procedure in [BWHT07].

Finite elements were also used for fracture in [MMDJ01], For isotropic material, their approach can be performed in real-time by neglecting some transient behavior of stiff materials. Other work includes the adaptive framework of [DDCB01], which uses tetrahedra on coarse mesh simulation and surface is refined according to sampling. Their adaptive technique is applied on model, space and time. The rotation based approach are explained in [MDM$^+$02] and [MG04], Müller et al. extract the rotational part per node which is different from existing approaches extracted per element [MDM$^+$02], the global stiffness matrix does not need to be reassembled at each time step. To eliminate artifacts, Müller et al. extract the rotational part of the deformation for each finite element and compute the forces with respect to the non-rotated reference frame [MG04]. The hybrid finite element free form deformation approaches such as [CGC$^+$02a, CGC$^+$02b], each region of the finite element mesh builds mapping relationship with the bone of a simple skeleton and then locally linearized. The regions are blended in each time step, leading to results which are visually acceptable and achieves better performance. Finite volume muscle models can be studied on [TBHF03].

2.2 Mass-spring system

Mass-spring systems might be the simplest and intuitive model compared with all other physical-based deformable models. Instead of using a PDE and discrete in space such as finite element method. Mass-spring systems handles a discrete model. The models are expressed by mass points which is connected by a network of massless springs. Physical properties are carried and propagated through springs.

With deformation energy being flowing through springs, the motion of each mass point is then governed by Newton's second law $F_i = k_i x_i$. The entire mass spring system can be expressed as

$$M\ddot{x} = f(x, v) \tag{2.2}$$

Time step is defined as t and x_i means the position of mass points $i = 1, ..., n$. The force f_i on each mass point is computed according to its neighbors under the spring network. where M is a 3n × 3n diagonal mass matrix. The system can be solved via a numerical integration scheme.

Earlier works of mass-spring networks are first being used for facial modeling [PB81, Wat87] in Computer Graphics, which are solving static problems. Dynamic models were later introduced to model skin and muscle [CHP89, WT91].

2.3 Meshless method

Meshless methods are capable of simulating a wide variety of materials and phenomena for their separately rendering process. Simulation is not constrained through particles simulation and mesh surface is reconstructed according to simulation results. So properties on mesh surface will not constrain particle simulation. Material behaviors are expressed by relationship between particles. Early work in meshless methods was done by Desbrun and Gauscel [DpG95], who calculated elastic forces using dynamically determined neighborhoods to allow behavior similar to plastic flow. Müller et al. use particles carrying strain and stress energy during simulation [MKN$^+$04]. For small plastic flow, they store plastic strain and then remove it before computing elastic forces. For large plastic flow, it stores and maintains elastic strain instead of plastic strain. The advantage of using meshless approach is that relationship of particles can be dynamically changed for general material usage.

2.4 Multi-resolution techniques

Multi-resolution techniques based on the structure of decomposing mesh into a low frequency base mesh with decreasing level of detail, geometric high frequency details are stored during decomposing. When base mesh deforms, Details are reconstructed according to its decomposing rule and get final deformation result. Focus on different deformation issues, decomposition and reconstruction techniques are different from approaches.

Lee introduce displacement vector on subdivision surface to reconstruct original mesh from smooth domain surface [LMH00]. Mesh is simplified and displacement vector is stored according to simplified vertex and new generated surfaces. When coarse mesh deforms, resolution is chosen from subdivision routine and displacement vector is applied on subdivision surface to get original mesh deformation. Botsch uses volume elements enclosed between the different resolution level to encode the detail information [BK03]. Keeping these displacement volumes locally constant during a deformation of the base surface leads to a natural behavior of the detail features. The reconstruction is a hierarchical iterative relaxation scheme, providing interactive response. Marinov present an efficient technique for multi-resolution deformations on GPU [MBK07]. These above methods requires subdivision connectivity. Kobbelt generalize multi-resolution techniques without requiring subdivision connectivity [KCVS98]. It stores mesh simplification information and use discrete fairing on mesh smoothing. However, These methods all focus on decomposing and resulting reconstruction. The multi-resolution control is a globally remesh structure. Detailed resolution control such as locally mesh refinement are not mentioned.

2.5 Differential coordinate

Lipman first introduced differential coordinate to interactive mesh editing [LSC$^+$04]. It represents differential coordinate by relationship between a vertex and its neighborhood region. Once mesh is edited by user, fixing some vertex positions and solving a least-square system to reconstruct final deformed mesh. The idea is similar to [SCO04] which discusses mesh reconstruction according to its connectivity. The more vertex position constraints are set, the more approximation compared to original mesh. By pre-computing pseudo-inverse matrix. Lipman achieves interactive performance under fixed interested region he want to edit [LSC$^+$04]. However, it is limited on pre-computation time of pseudo-inverse matrix calculation. Once user changed interested region he want to edit. It takes a few seconds to recompute the pseudo-inverse matrix. The time complexity of calculating pseudo-inverse matrix is $O(n^2)$. Based on the same structure, Sorkine focus on a better representation of differential coordinate [SLC$^+$04]. It computes an appropriate transformation matrix for each vertex based on the eventual new configuration of deformed vertices. Lipman introduce a rigid motion invariant mesh representation based on discrete forms defined on the mesh [LSLC05]. The reconstruction of mesh geometry requires solving two sparse linear systems. First solving relationship between local frames and then solving position of the vertices via local frames. Zhou extended differential coordinate representation to volume [ZHS$^+$05]. It construct a graph representing the volume inside the input mesh. The graph's Laplacian encodes volumetric details as the difference between graph points and its neighbors. Volume relationship and constraints are added into optimization scheme. Nealen compared different differential coordinate representation and discussed their strength and weakness [NISA06]. These above works follow the whole optimization structure and are limited in pre-computation time, lose control in generality on dynamic usage.

Focus on performance advancement, Wei introduced a subspace technique. Build a coarse control mesh around the original mesh and project the deformation energy and constraints onto control mesh vertices. The energy minimization is then carried out in subspace. Vertices deformation on original mesh are interpolated using mean value interpolation according to subspace optimization results [HSL+06]. Sumner used similar structure by introducing deformation graph [SSP07]. Original mesh is embedded into a deformation graph. Optimization is applied on deformation graph and original mesh deformation are interpolated by graph results. Lipman [LLCO08] and Langer [LS08] introduce interpolation coordinates that preserves detail better compared to mean value interpolation used in [HSL+06]. These subspace structures achieved dynamic interactive performance. However, subspace explained above are both a coarse mesh which contains no detail as shown in Figure 2.1. Optimization are only applied on coarse control mesh to catch a global appearance. It is hard to claim detail-preserving because details are not reconstructed through optimization but through interpolation. Because details are reconstructed through interpolation, the subspace structure lose control on local detail. Deformation resolution is limited on resolution of coarse control mesh.

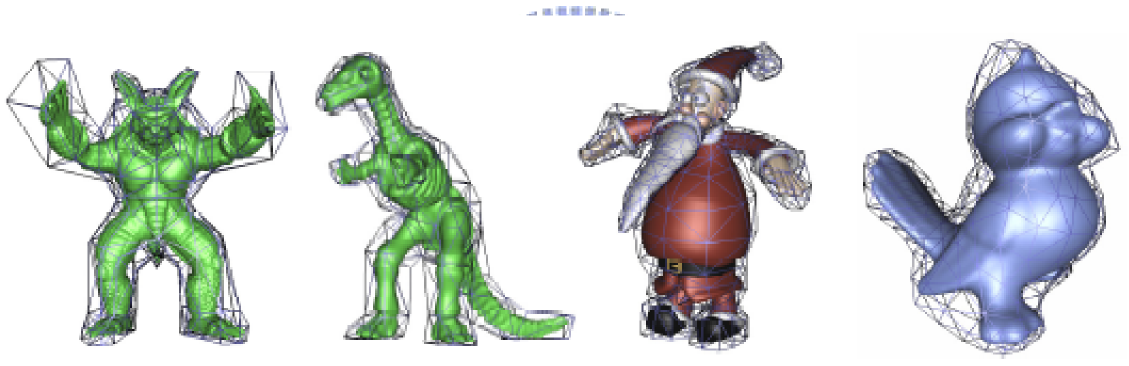


Figure 2.1: Coarse control meshes around the original fine meshes [HSL+06].

Our algorithm is based differential coordinate. Now we introduce some background of differential coordinate: Differential coordinate is a representation on mesh detail property. It encodes relationship between a mesh vertex and its neighbors. Upon deformation, solving a least-square optimization on vertices to maintain global appearance and local detail property.

2.5.1 Representation

Lipman first introduced differential coordinate to interactive mesh editing [LSC+04]. Let $G = (V, E)$ represent a triangular mesh on 3D, where V denotes the set of vertices of the mesh and

E denotes the set of edges. Mesh topology is implied on these information. The spatial position of vertex j is represented by p_j. Let S be a scheme approximating a vertex $p_j \in V$ by linear combining some other vertices:

$$p_j \approx S(p_j) = \sum_{i \in supp(j), i \neq j} \alpha_{ji} p_i \tag{2.3}$$

where $supp(j)$ denotes the set of vertex indices that approximate vertex j by scheme S.

Mesh Laplacian operator is a simplest linear differential mesh operator created by scheme S [SLC$^+$04]. It is represented by vertex and its one ring neighborhood. Let the mesh M be described by a pair (K, V), where K describes the connectivity and V = $\{v_1, \ . \ . \ . \ , v_n\}$ describes the geometric vertex positions in R^3. The neighborhood of a vertex i is the one ring adjacent vertices N_j = { j | $(i, j) \in K\}$. Laplacian coordinate defines as follows:

$$\begin{aligned} \delta_i &= L(v_i) \\ &= v_i - \frac{1}{d_i} \textstyle\sum_{j \in N_i} v_j \end{aligned} \tag{2.4}$$

where δ_i is the Laplacian coordinate of vertex i, v_i is the position of vertex i and d_i is the degree of v_i.

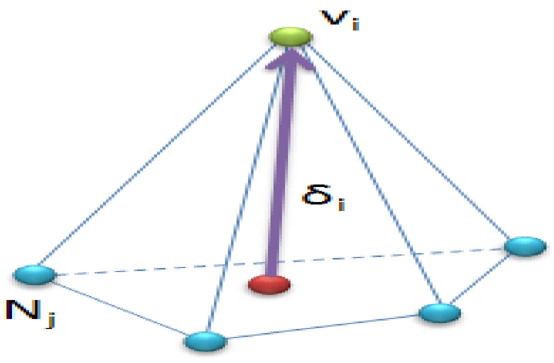
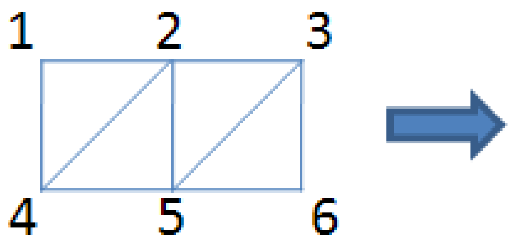


Figure 2.2: Laplacian coordinate for vertex i.

Laplacian coordinate encodes positional velocity difference between a vertex position and the average of its one ring neighborhood vertices' position as show in Figure 2.2. Let A be the mesh adjacency matrix and D = diag($d_1, \ . \ . \ . \ ,d_n$) be the degree matrix. Then a set of



$$L=\begin{bmatrix} 1 & -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 \\ -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 0 \\ 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 1 & -\frac{1}{4} \\ 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \end{bmatrix}$$

Figure 2.3: Laplacian operator L.

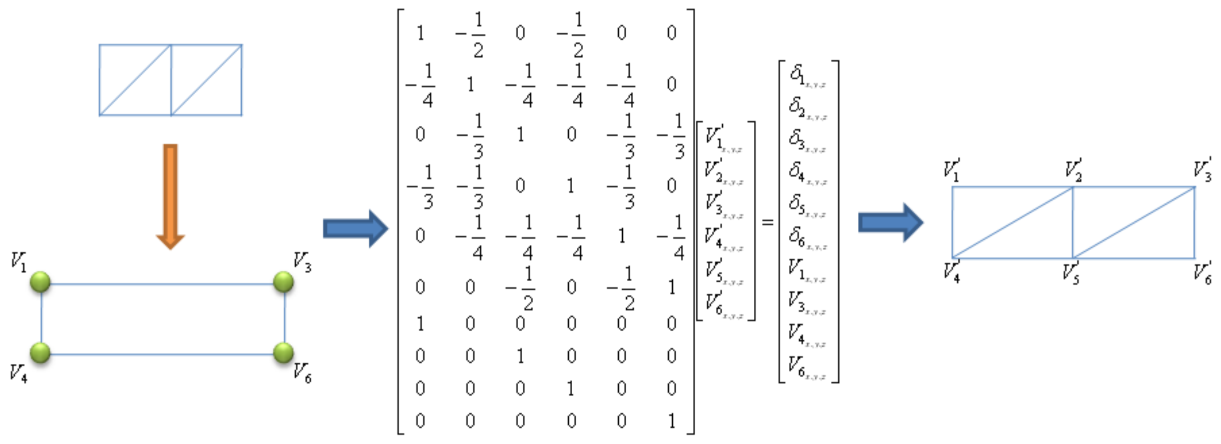
differentials $\Delta = \{\delta_i\}$ can be expressed as $\Delta = LV$, where $L = I - D^{-1}A$. The matrix L is the Laplacian operator of the mesh with connectivity A(Figure 2.3).

2.5.2 Optimization

To perform deformation using Laplacian coordinate Δ is to fix the absolute position of several vertices and solve for the remaining vertices by fitting the Laplacian coordinates of the geometry to the given Laplacian Δ. The result is solved by using least-square approach in the following error functional:

$$E = minimize \sum_{i=1}^{n} \parallel \delta_i - L(v_i^{'}) \parallel^2 + \sum_{i=m}^{n} \parallel v_i^{'} - u_i \parallel^2 \tag{2.5}$$

Absolute position of several vertices $v_i^{'} = u_i$, $i \in$ {m,...,n}, m $<$ n. The rationale of fitting given Laplacian coordinates is that details of the shape are preserved, as the relative location of vertices is encoded in Δ. An example is show in Figure 2.4.



$$\begin{bmatrix} 1 & -\frac{1}{2} & 0 & -\frac{1}{2} & 0 & 0 \\ -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 0 \\ 0 & -\frac{1}{3} & 1 & 0 & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 0 & 1 & -\frac{1}{3} & 0 \\ 0 & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 1 & -\frac{1}{4} \\ 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V'_{1_{x,y,z}} \\ V'_{2_{x,y,z}} \\ V'_{3_{x,y,z}} \\ V'_{4_{x,y,z}} \\ V'_{5_{x,y,z}} \\ V'_{6_{x,y,z}} \end{bmatrix} = \begin{bmatrix} \delta_{1_{x,y,z}} \\ \delta_{2_{x,y,z}} \\ \delta_{3_{x,y,z}} \\ \delta_{4_{x,y,z}} \\ \delta_{5_{x,y,z}} \\ \delta_{6_{x,y,z}} \\ V_{1_{x,y,z}} \\ V_{3_{x,y,z}} \\ V_{4_{x,y,z}} \\ V_{6_{x,y,z}} \end{bmatrix}$$

Figure 2.4: Example of solving equation 2.5.

CHAPTER 3

Detail Preserving Deformation

We proposed a detail-preserving deformation method to seek balance between performance and quality during runtime process. It is based on an intuitive idea: using base mesh deformation to drive original mesh deformation. The idea is widely used on some other works [BK03, CGC$^+$02a, CGC$^+$02b, KCVS98, MBK07]. Based on this idea, our system can be explained in Figure 3.1 and Figure 3.2.

In our system, an input original mesh is simplified to a base mesh at pre-processing stage. The criterion we choose for simplifying mesh is that we simplify *details* which are locally high frequency features on mesh. A hierarchical tree with dependency is constructed during simplification. Every node(except root) on the hierarchical tree represents a vertex removal/recover operator. A cut on the hierarchical tree represents a cut mesh (A coarser mesh compared to original mesh, as shown in Figure 3.2). Cut contains the root alone means base mesh and cut contains the whole tree nodes means original mesh. MAPS [LSS$^+$98] parameterization is built at this stage for initial rotation assumption of differential coordinate. During runtime process, we trace the hierarchy to find a cut seeking balance between performance and detail preserving quality. Directly recover method is applied on nodes above cut to get cut mesh deformation. Sub-region optimization scheme is applied on nodes below cut to get original mesh deformation as shown in Figure 1.2.

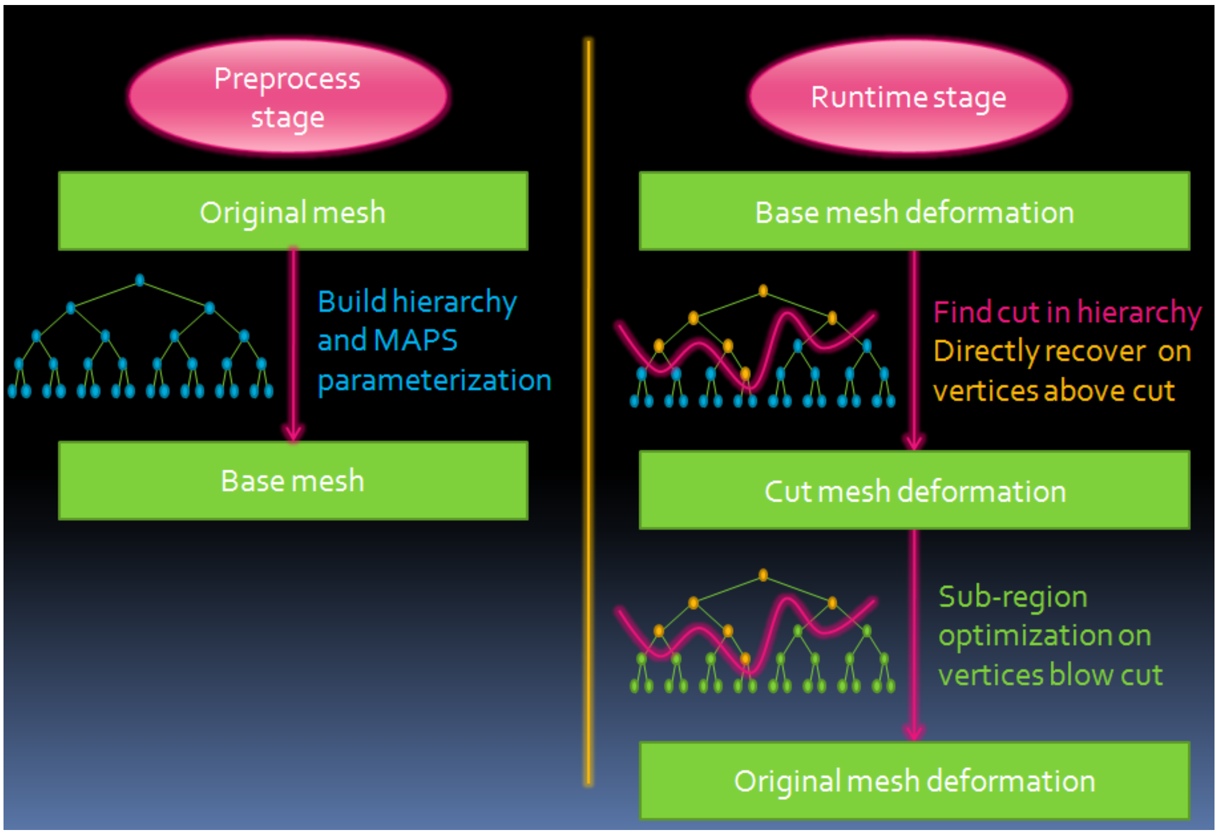

Preprocess stage
Original mesh
Build hierarchy and MAPS parameterization
Base mesh
Runtime stage
Base mesh deformation
Find cut in hierarchy
Directly recover on vertices above cut
Cut mesh deformation
Sub-region optimization on vertices blow cut
Original mesh deformation

Figure 3.1: System structure.

Our system is independent of coarse mesh deformation algorithms. We can support both physics-based methods such as finite element method(FEM), mass-spring system and non-physics-based methods, such as free-form deformation(FFD)(Figure 3.3) and skeleton-based mesh deformation(Figure 3.4).

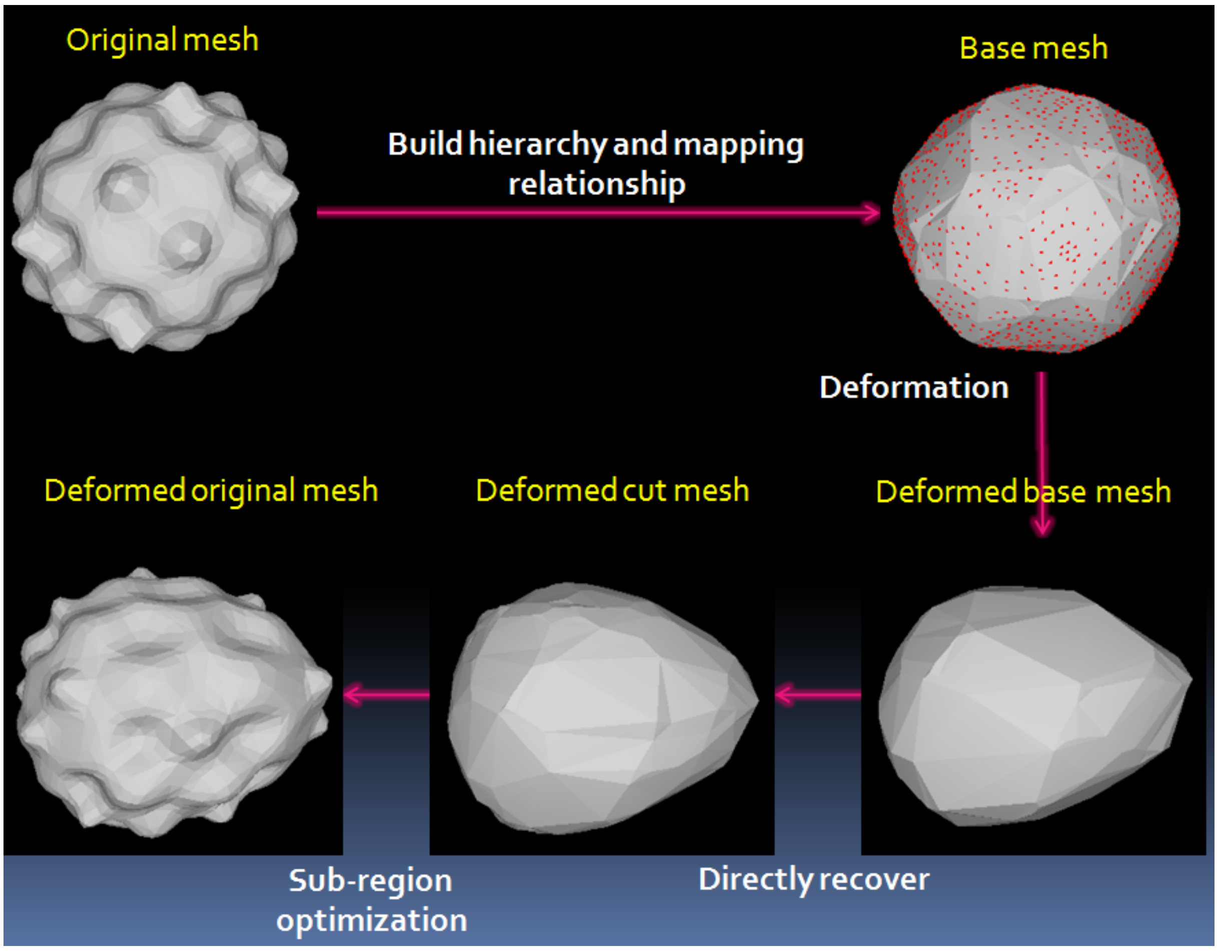
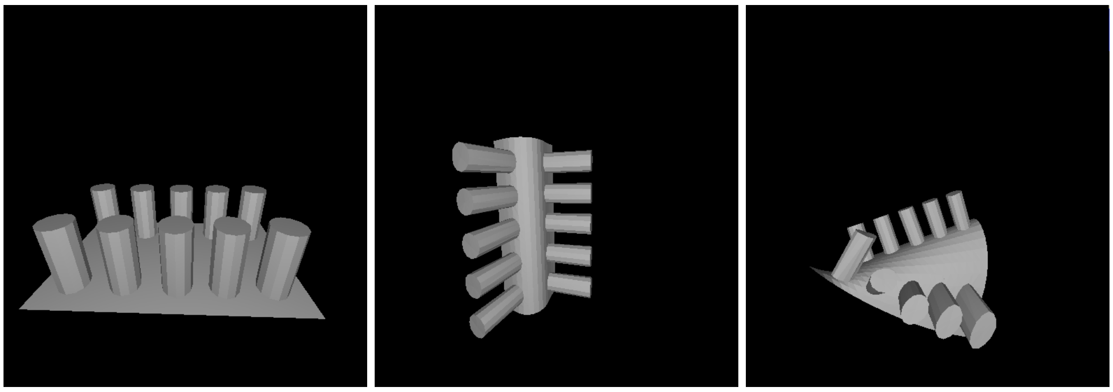


Figure 3.2: System structure example.

3.1 System Structure

Our system controls base mesh deformation to drive original mesh detail-preserving deformation with balanced performance and quality. It can be divided into two stages: preprocessing stage and dynamic simulation stage. At preprocessing stage, a hierarchy with dependency and mapping relationship between vertices on base mesh and original mesh are built. At dynamic simulation stage, finding a performance and detail-preserving quality balanced cut on hierarchy. Directly recover method are applied on vertices above the cut to catch global appearance. Sub-region optimization are applied on vertices below the cut to preserve detail.



(a) Original mesh. (b) Deformation based on our method. (c) Deformation based on our method.

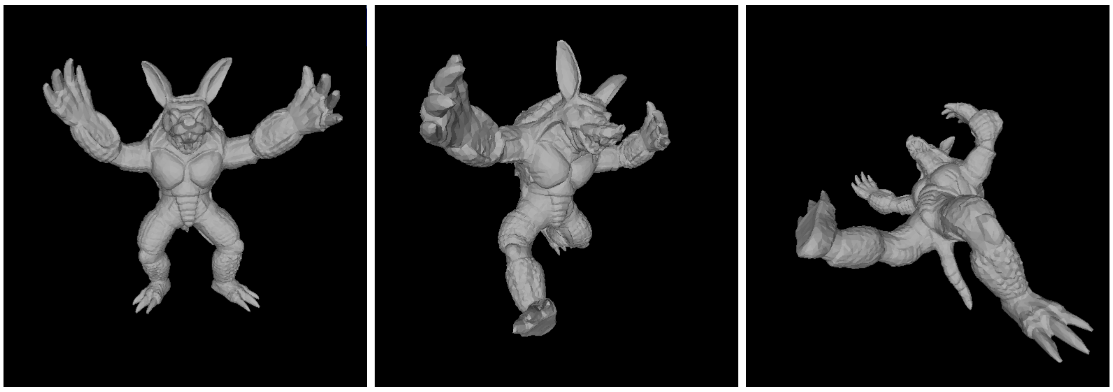
Figure 3.3: Our method applied on free-form deformation.

3.1.1 Build hierarchy

Hierarchy is built in order to increase performance without losing detail control. Directly recover method is applied on vertices above the cut to catch global appearance. Subregion optimization is applied on vertices below the cut to preserve detail. Because details should be optimized during deformation, detailed vertices must appear at bottom of the hierarchy. Simplification criterion is designed on simplifying locally high frequency details.

Original mesh is simplified to base mesh like a general progressive mesh approach [Hop96] as shown in Figure 3.5. Simplification criterion is designed not only for detail-preserving consideration but also for simulation consideration such as stability and convenience. For example, physical simulation using finite element method divides mesh into set of elements. Details on mesh will be approximated by tiny elements. They not only increase simulation complexity(more elements) but also decrease simulation stability(tiny element is easy to become ill-conditioned after deformation). Our method simplified mesh into base mesh with almost no detail. Simulation applied on base mesh is much easier and stable.

Compared to original mesh, base mesh only need to support a broadly deformation. So we tend to simplify locally high frequency mesh details. Because these details are less simulated



(a) Original mesh. (b) Deformation based on our method. (c) Deformation based on our method.

Figure 3.4: Our method applied on skeleton-based deformation.

and they are comparable less important and uninfluential for the whole broadly mesh deformation. To determine mesh details, we consider curvature [MDSB02] and mesh saliency [LVJ05]. We use mean curvature normal operator to express curvature and a scale-dependent manner using a center-surround operator on Gaussian-weighted mean curvatures to express mesh saliency. The higher vertex curvature is, the more we claim as details and we tend to simplify the vertex. Mesh saliency vertices should be kept during simplification because they are features during simulation. An example is shown in Figure 3.6.

We choose vertex removal as simplification operator(half-edge collapse is also supported). A vertex is chosen to be removed according to simplification criterion, along with its adjacent edges and triangles. Triangulation is then applied on the resulting hole as shown in Figure 3.7. Vertices' simplification cost are calculated and inserted into a priority queue with lazy update approach [CMO97], which uses greedy approach to avoid recalculating cost by setting a dirty flag on topology changed vertices. The cost of selected simplification vertex is recalculated if the dirty flag is set and push into the priority queue. During simplification, we construct a dependency hierarchy tree. A tree node represents a vertex removal/recover operation. We choose vertex removal/recover operation as tree node and build dependency relationship on parent and child relationship. To build hierarchy contained dependency, when simplifying a

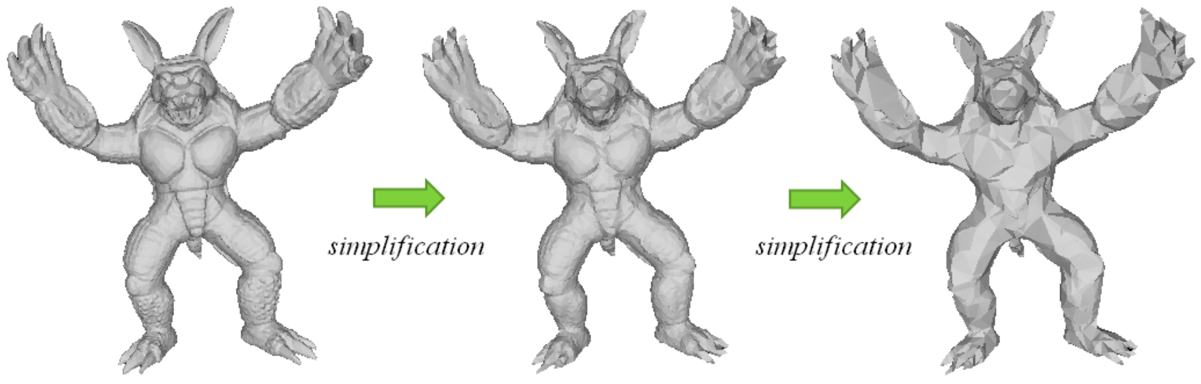


Figure 3.5: Level of detail example on Armadillo model.

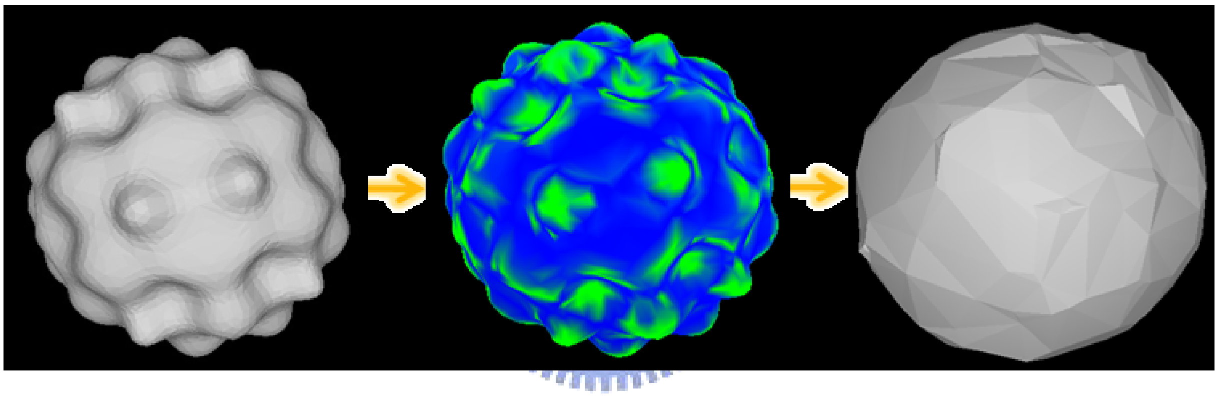


Figure 3.6: Mesh simplification example. Green parts are claimed as details.

vertex, a node is added on the tree and label one ring neighborhood vertices connection edges and new generated edges with node index as shown in Figure 3.8. Labeling an edge means the edge is affected by vertex removal. During each simplification operation, if removal edges are labeled, the new tree node we add to the tree will be added as parent of the whole labeled index related nodes as shown in Figure 3.9, which means edge is affected by previous vertex removal operations(for example: edge is new generated by vertex removal) and there should be an order when we simplify or recover vertices in hierarchy. Simplification/recover order dependency is implied on parent-child relationship. A node can be recovered only if all its parents are recovered already and a node can be removed only if all its children are removed

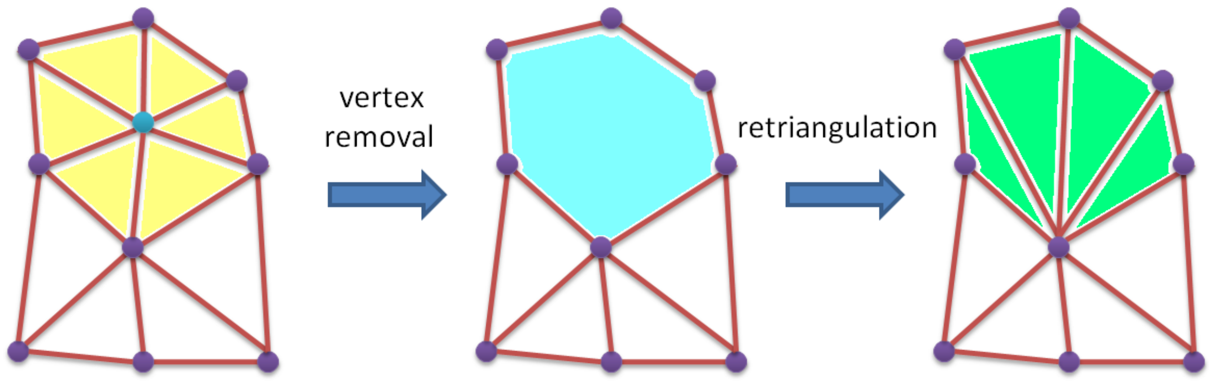
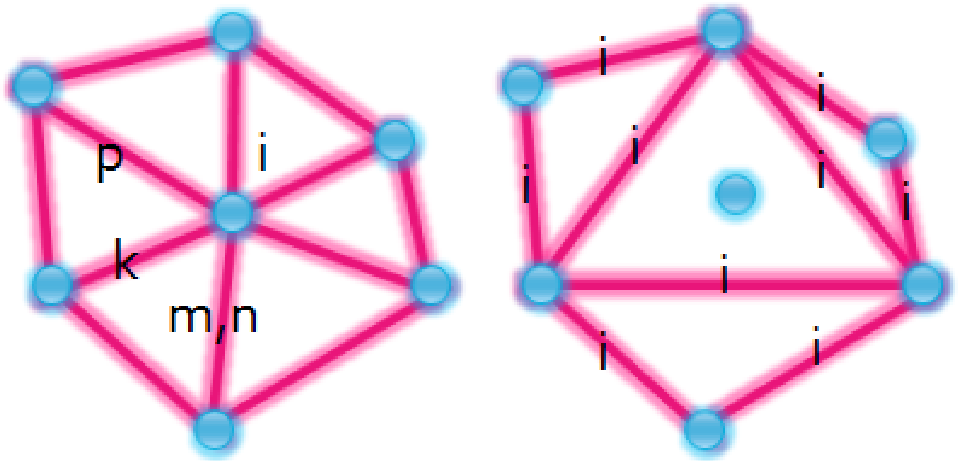


Figure 3.7: A vertex removal operation.

already. Dependency problems are now easy to handle by checking valid parents or children. An detailed example is shown in 3.10. The hierarchy imply simplification order through edge label checking and encoded on parent-child relationship. So each sub-tree in hierarchy forms an independent region.

Following the dependency hierarchy construction rule, when simplification ends we have dependency forests. Each tree contains connected vertices that form a region in original mesh. We use a root node to connect all naked nodes(nodes without parents) and finish constructing a tree with dependency. Each node on the tree is a vertex removal/recover operation. A cut on the tree means a cut mesh(coarser mesh compared to original mesh) and its topology. A node contained in the cut means the node index incident vertex is contained in current cut mesh. A cut contains whole tree nodes means original mesh(no vertices are removed). A cut contains only root vertex means base mesh. To traverse the tree to find a cut, top-down a node means recovering a vertex and update topology information, bottom-up a node means removing a vertex and update topology information.



(a) Simplified vertex and its one ring condition before vertex removal.

(b) Labeled edges after vertex removal.

Figure 3.8: Label condition after vertex removal.

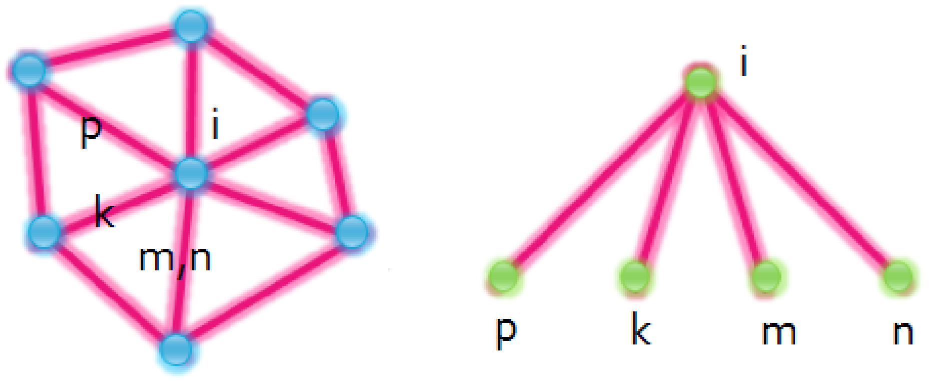
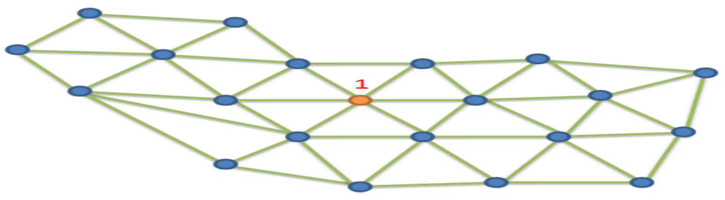
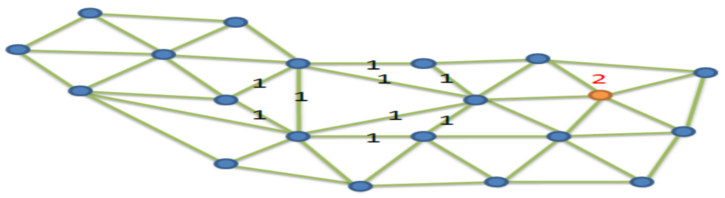


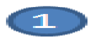
Figure 3.9: Tree construction.



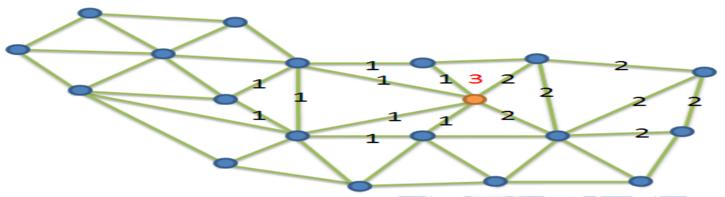
(a) Mesh



(b) Vertex removal.

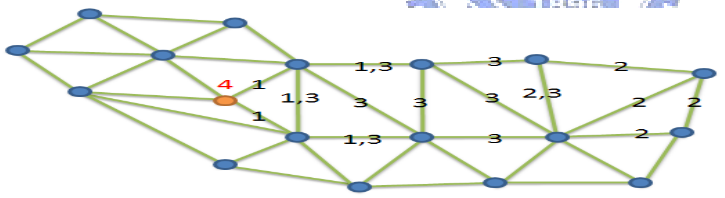


(c) Build hierarchy.

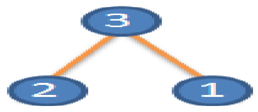


(d) Vertex removal.

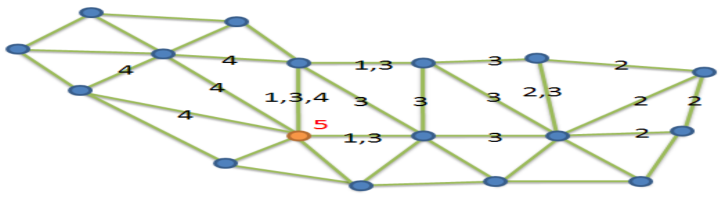
(e) Build hierarchy.



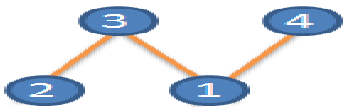
(f) Vertex removal.



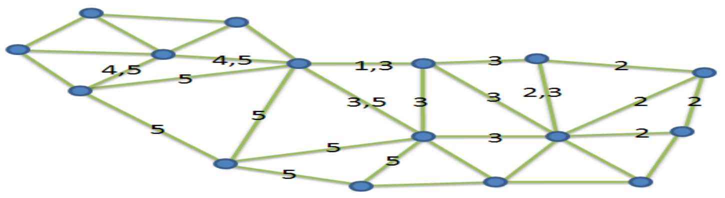
(g) Build hierarchy.



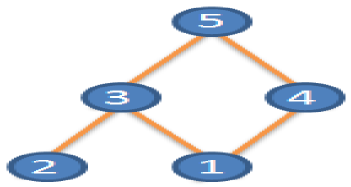
(h) Vertex removal.



(i) Build hierarchy.



(j) Vertex removal.



(k) Build hierarchy.

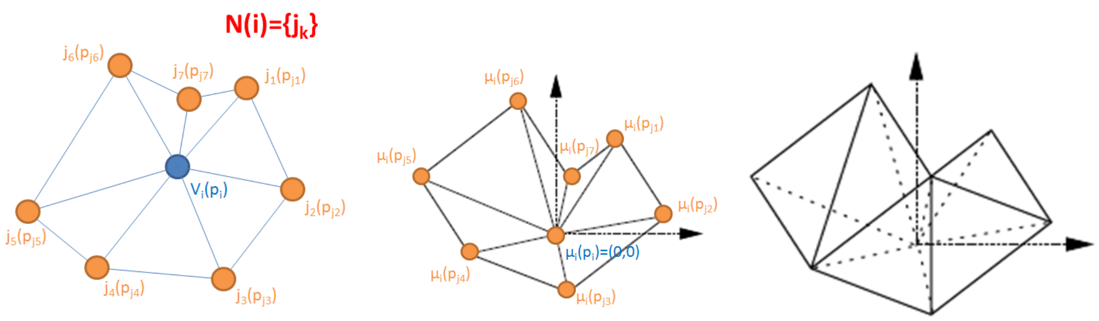
Figure 3.10: Vertex removal and building hierarchy.

3.1.2 Build mapping relationship

Mapping relationship is built between original mesh and base mesh, which is built for rotational invariance assumption of differential coordinate to help iterative optimization convergence. We choose MAPS parameterization [LSS$^+$98]: Each vertices on original mesh can be expressed as barycentric coordinate of base mesh vertices. We extract rotation information from base mesh deformation and calculate rotation assumption on original mesh vertices through quaternion barycentric interpolation.

MAPS parameterization is built during mesh simplification: When a vertex is removed, the naked area is re-triangulated and removed vertex is parameterized into one of the new generated triangles. To triangulate the naked area, first we project one ring neighborhood vertices into 2D area using conformal mapping. Let $\{i\}$ be a vertex to be removed. Enumerate cyclically the K_i vertices in the one ring $N(i) = \{j_k\}$ such that $\{j_{k-1}, i, j_k\}$ is a removed triangle. A linear approximation which we denote by μ_i is defined by its values for the center point and one ring neighbors; namely $\mu_i(p_i) = 0$ and $\mu_i(p_{j_k}) = r_k^a exp(i\theta_k a)$, where $r_k = \| p_i - p_{j_k} \|$, $\theta_k = \sum_{l=1}^{k} \angle(p_{j_{l-1}}, p_i, p_{j_l})$ and $a = \frac{2\pi}{\theta_{K_i}}$ Now we have a conformal mapped 2D closed area with removed vertex projected on $\in (0,0)$ in 2D and we triangulate the area as show on Figure 3.11. 2D triangulation results are directly mapped onto 3D. The mapping of a point p can be written as $p = \alpha p_i + \beta p_j + \gamma p_k$ where $\{i, j, k\}$ is a new created face and α, β and γ are barycentric coordinates, i.e., $\alpha + \beta + \gamma = 1$ (Figure 3.12). After simplification ends, all removed vertex are parameterized into a base domain triangle with barycentric coordinates.

Rotation information can be extracted from base mesh deformation by calculating local coordinate system transformation on each vertex. As shown in Figure 3.13, local coordinate system is constructed for each vertex i on base mesh. N_i is normal on v_i, E_i is the one ring outgoing edge of v_i, the angle between N_i and E_i is minimal. N_i, $N_i \times E_i$ and $(N_i \times E_i) \times E_i$ formed an orthonormal basis of the local coordinate system L.



(a) Vertex topology in 3D. (b) Mapping into 2D under conformal mapping. (c) Corresponding 2D triangulation.

Figure 3.11: 2D triangulation using MAPS [LSS$^+$98].

Upon deformation, rotation matrix can be extracted by

$$\begin{aligned} L_{1,i}R_i &= L_{2,i} \\ R_i &= (L_{1,i})^{-1}L_{2,i} \end{aligned} \tag{3.1}$$

$L_{1,i}$ is the original local coordinate system formed by v_i and $L_{2,i}$ is the updated local coordinate system after deformation. Represent rotation matrix by quaternion and use MAPS parameterized barycentric coordinates, an initial rotation assumption on original mesh vertices can be calculated through quaternion barycentric interpolation.

3.2 Differential coordinate representation

Our differential coordinate representation extend the idea on Huang's research [HSL$^+$06]. Huang observed that Laplacian is a discrete approximation of the curvature normal and cotangent form Laplacian lies exactly in the linear space spanned by the normals of the incident triangles. From these observations, Their differential coordinate representation can be expressed as

$$d_i(X) = \Sigma_{j=1}^{n_i}\mu_{ij}((x_{i,j-1} - x_i) \times (x_{i,j} - x_i)) \tag{3.2}$$

where $d_i(X)$ is the differential coordinate of vertex x_i on original mesh, $x_{i,1}$,...,x_{i,n_i} are the one ring neighborhood vertices of x_i, $\times$ expresses the cross product of two vectors in R^3, so

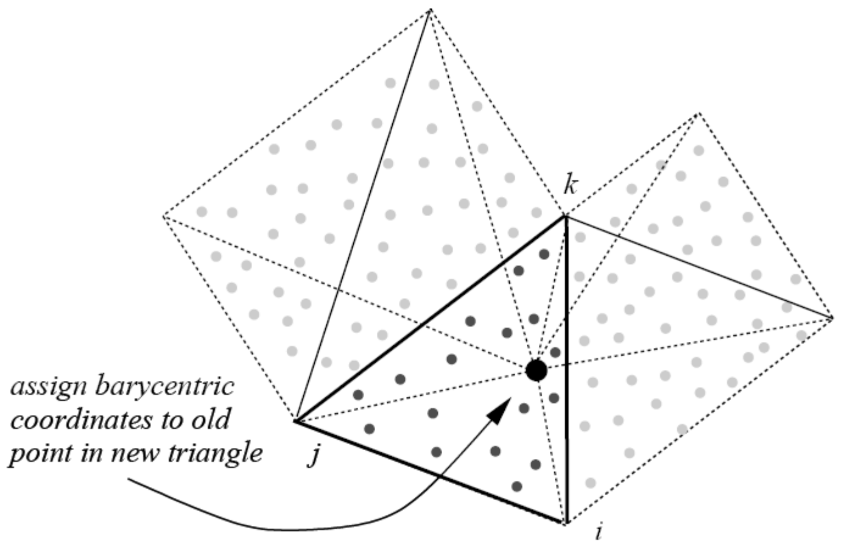


Figure 3.12: MAPS Parameterization [LSS+98].

the term $(x_{i,j-1} - x_i) \times (x_{i,j} - x_i)$ indicates the normal of incident triangles. Since $\mu_i j$ are constant coefficients which remain invariant with respect to rigid rotation. $d_i(X)$ provides a rotation-invariant differential coordinate representation.

The set of coefficients $\mu_i j$ are solved by an under-constrained system. Let A_i be the $3 \times n_i$ matrix whose j-th column is $(x_{i,j-1} - x_i) \times (x_{i,j} - x_i)$ and μ_i be $(\mu_{i,1}, ..., \mu_{i,n_i})$. Then $d_i(X) = A_i \mu_i$. Huang solve μ_i by computing the pseudo inverse A_i^+, through singular value decomposition and set $\mu_i = A_i^+ d_i(X)$ [HSL+06]. It is equivalent to finding a solution on 3.2 that minimizes $\|\mu_i\|$.

Based on Huang's observation. We found that cotangent Laplacian

$$\begin{gathered} \delta_i = v_i - \frac{1}{\Sigma_{(i,j)\in E} w_{ij}} \Sigma_{(i,j)\in E} w_{ij} v_j \\ w_{ij} = \cot\theta_{ij} + \cot\gamma_{ij} \end{gathered} \tag{3.3}$$

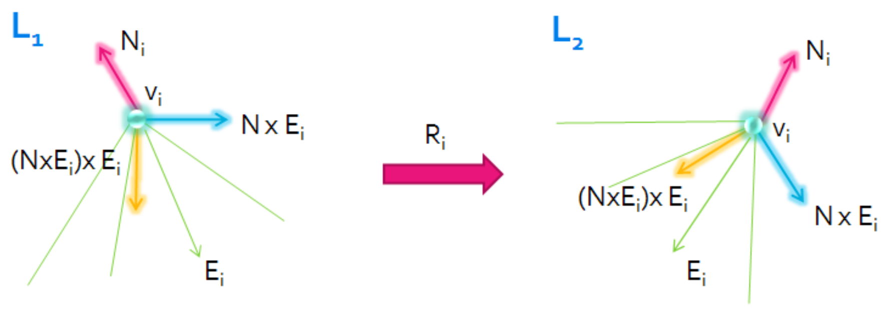


Figure 3.13: Local coordinate system transformation.

can be rewritten as

$$\begin{aligned} \delta_i &= v_i - \frac{1}{\Sigma_{(i,j)\in E} w_{ij}} \Sigma_{(i,j)\in E} w_{ij} v_j \\ \Sigma_{(i,j)\in E} w_{ij}\delta_i &= \Sigma_{(i,j)\in E} w_{ij} v_i - \Sigma_{(i,j)\in E} w_{ij} v_j \\ &= \Sigma_{(i,j)\in E} w_{ij}(v_i - v_j) \\ &= 2 \times Amixed \times K(v_i) \\ &= 2 \times Amixed \times C_i \times n_i \\ &= 2 \times Amixed \times C_i \times \Sigma_{j=1}^{n_i} \lambda_{ij}((x_{i,j-1} - x_i) \times (x_{i,j} - x_i)) \end{aligned} \tag{3.4}$$

$K(v_i)$ is the mean curvature normal operator [MDSB02], C_i means the curvature on vertex i, calculation of Amixed can also be found on [MDSB02]. Vertex normal can be approximated by incident triangle normals, we choose $\lambda_{ij} = \frac{T_j}{\sum_{j\in N_i} T_j}$, T_j expresses the incident triangle area.

Based on the assumption to preserve curvature, we rewrite 3.2

$$d_i(X) = \Sigma_{j=1}^{n_i} \rho_{ij}((x_{i,j-1} - x_i) \times (x_{i,j} - x_i)) \tag{3.5}$$

with unique weighting ρ_{ij} for each vertex i.

3.3 Cut updating criterion

To dynamically update cut, we consider deformation properties to catch acute deformation and cut corresponding regions' boundary to maintain smoothness on deformation after optimization.

Cut updating criterion is designed as

$$w_1 \sum \Delta L^2(T) + w_2 \sum ND_E \tag{3.6}$$

A vertex recover applied on cut mesh means increased performance and decreased quality(Optimization on fewer vertices). Likewise, a vertex removal applied on cut mesh means decreased performance and increased quality. Each valid vertex removal operation cost(above cut) and vertex recover operation cost(below cut) are inserted into two different priority queues. Finding pair operations(each pair contains a vertex removal operation and a vertex recover operation on the two priority queues) with increased performance and quality to update cut.

3.3.1 Geometric stretch

To catch deformation properties, we choose geometric stretch calculated on mesh triangle to evaluate deformation degree. Optimization must be applied on where acute deformation happens.

Geometric stretch L^2 and L^∞ can be calculated

$$\begin{gathered} L^2(T) = \sqrt{\Gamma^2 + \gamma^2}/2 \\ L^\infty(T) = \Gamma \end{gathered} \tag{3.7}$$

A triangle T with original 3D coordinates p_1, p_2, p_3 and corresponding deformed 3D coordinates q_1, q_2, q_3. There exists an unique affine mapping $S(p) = q$. We calculate Jacobian of the mapping S, Γ and γ represents the larger and smaller singular values of the Jacobian as shown in Figure 3.14. The norm $L^2(T)$ corresponds the root-mean-square stretch and $L^\infty(T)$ means the greatest stretch.

3.3.2 Normal difference

To preserve region boundary smoothness, we choose normal difference calculated on mesh edge. Because optimization is applied on each region independently, gaps might appear on

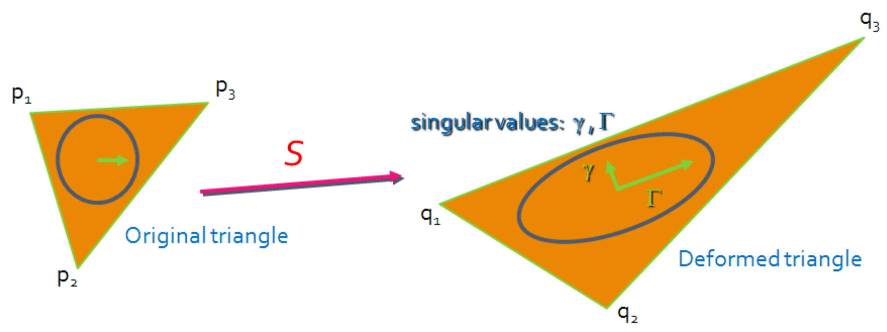


Figure 3.14: Geometric stretch.

region boundaries after optimization. To avoid these problem, we tend to choose region with less deformation on boundary for further optimization.

Normal difference represents orientation change between edge incident triangles. It is used to detect rotational deformation such as bending which might not be detected by geometric stretch. Normal difference is represented as

$$\begin{aligned} ND_E &= |E|\Delta\theta \\ \Delta\theta &= \theta_d - \theta \end{aligned} \tag{3.8}$$

As shown in Figure 3.15. $|E|$ means edge length, θ represents angle between edge incident triangle normals, θ_d is the same angle after deformation, $\Delta\theta$ represents change of the angle.

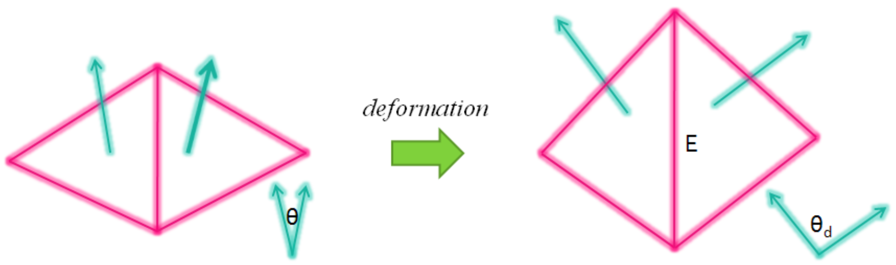


Figure 3.15: Normal difference.

Normal difference is calculated on region boundary edge. An example is shown in Figure 3.16.

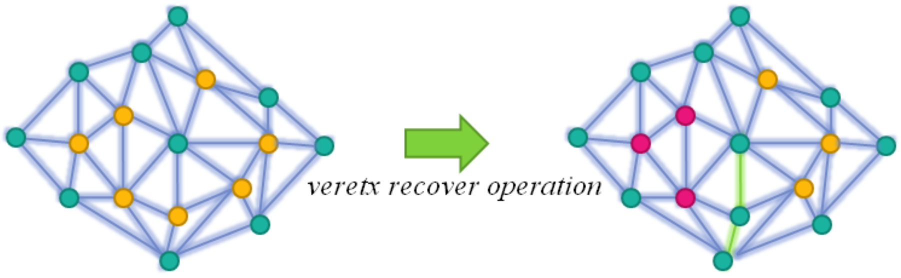


Figure 3.16: Vertex recover operation divide one region into two. Green edges are new appeared region boundary.

3.3.3 Performance threshold

Since optimization is the performance bottleneck on our system as shown in Figure 3.17. We set a threshold to control performance upper bound. Our performance estimation is designed by square sum of region vertex numbers:

$$T = \sum_{i=1}^{k} n_i^2 < THRESHOLD \tag{3.9}$$

n_i represents vertex number of region i. k means total region number. We choose n_i^2 because time complexity of solving least-square is $O(n^2)$. An example is shown in Figure 3.18.

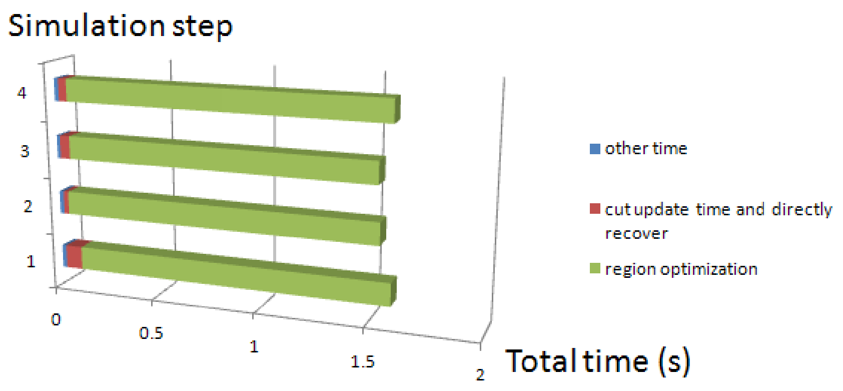


Figure 3.17: Performance calculation.

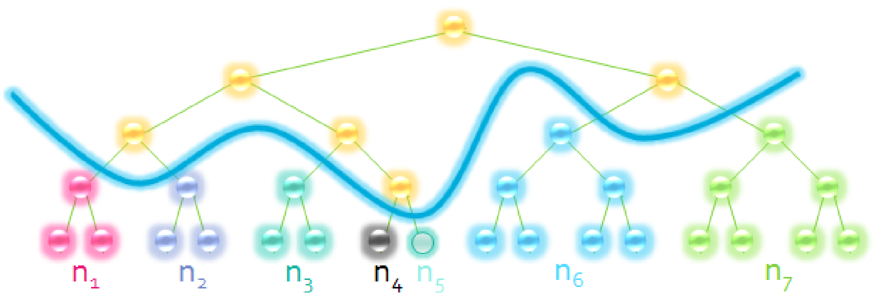


Figure 3.18: Performance estimation: $T = 3^2 + 3^2 + 3^2 + 1^2 + 1^2 + 7^2 + 7^2$.

3.4 Directly recover

Directly recover updates vertex position according to hierarchical sequential vertex recover operation. Differential coordinate is calculated on each vertex removal operation. After deformation, vertex is reconstructed through updated differential coordinate.

$$V'_{removal_i} = \delta'_{removal_i} + \frac{1}{\sum_{j \in N_i} w_{ij}} \sum_{j \in N_i} w_{ij} V'_j \tag{3.10}$$

A 2D example can be shown in Figure 3.19.

3.5 Region optimization

Region optimization is applied on hierarchical tree nodes below the cut. Each region is optimized according to Equation 2.5 with constrained vertices condition. Constrained vertices are boundary vertices or some inside vertices, which are vertices on cut mesh as shown in Figure 3.20. So each region can be optimized independently. An example is shown in Figure 3.21.

The optimization scheme of solving least-square on Equation 2.5 is the performance bottleneck. Its time complexity is $O(n^2)$ with mesh vertex number n. Region optimization is used to increase performance. Under Roughly evaluation, The calculation time T of solving least

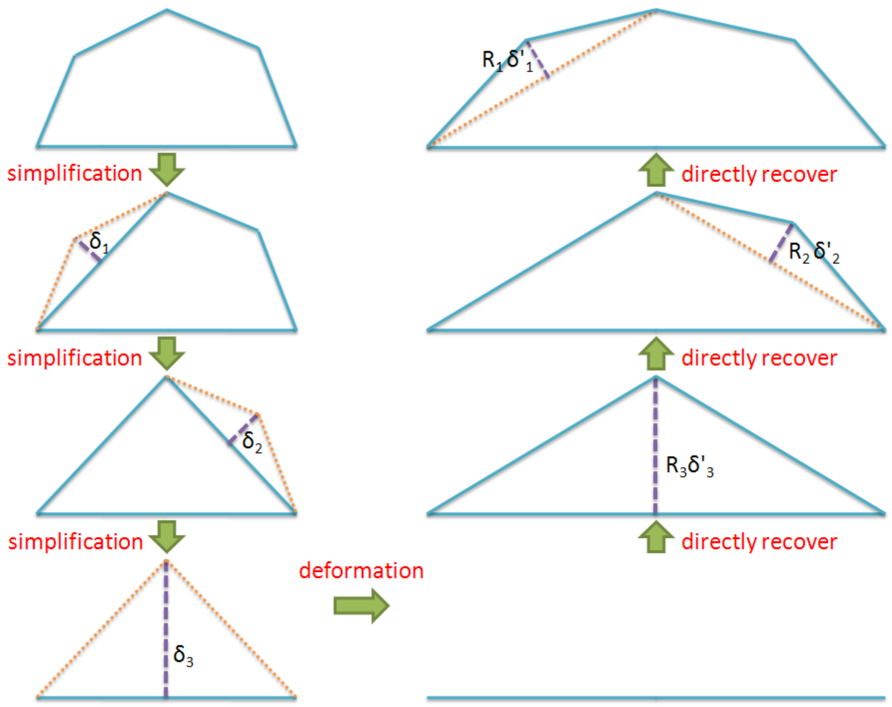


Figure 3.19: Directly recover example.

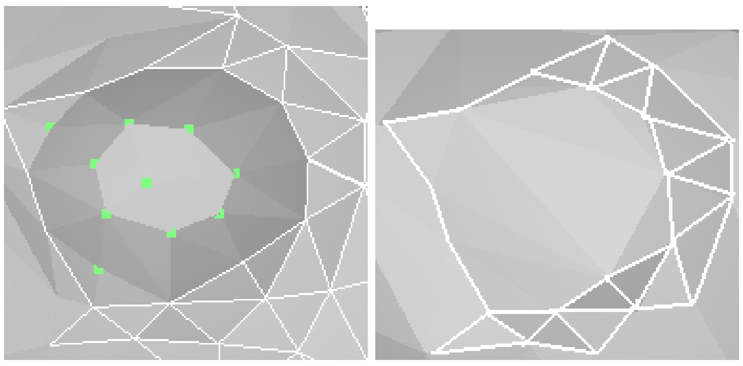
square will become $\frac{T}{k}$ if we can determine k independent regions as shown in below.

$$\begin{aligned} T &= O(n^2) \\ T &= sn^2 \\ T' &= s \times k \times (\tfrac{n}{k})^2 = \tfrac{sn^2}{k} \\ T' &= \tfrac{T}{k} \end{aligned} \tag{3.11}$$

To be more precisely, we categorize region vertices into optimized vertices and boundary constraints. Optimized vertices are vertices below cut and boundary constraints are cut mesh vertices which represents region boundary as shown in Figure 3.20. The performance calculation can be represent as

$$T = \sum_{i=1}^{k}(n_i^2 + c_{ni}^2) \tag{3.12}$$

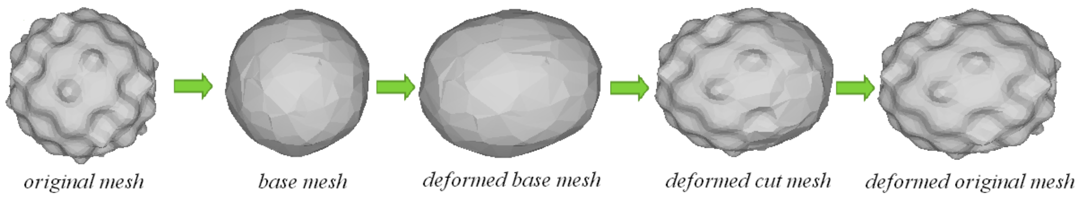
n_i represents optimized vertex number and c_{ni} represents constrained vertex number of region i. k means total region number.



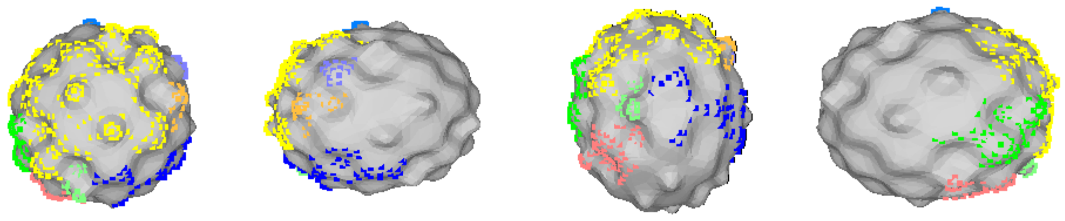
(a) Optimized region and boundary on original mesh. (b) Boundary constraint on cut mesh.

Figure 3.20: Region optimization with boundary constraint.

Region optimization not only increases performance but also preserves feature better by avoiding smoothing features vertices during whole optimization.



(a) A deformation example.



(b) Region optimization. Regions are displayed by different colors.

Figure 3.21: Region optimization example

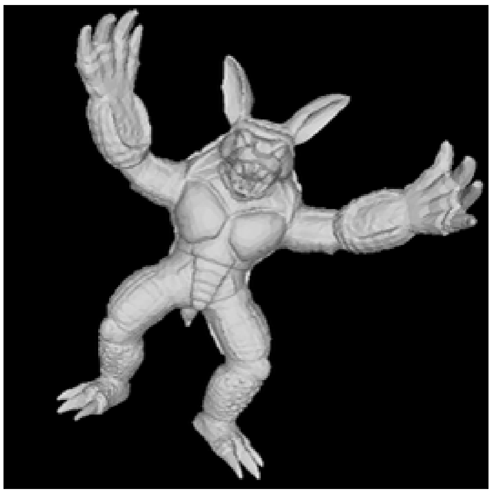
CHAPTER 4

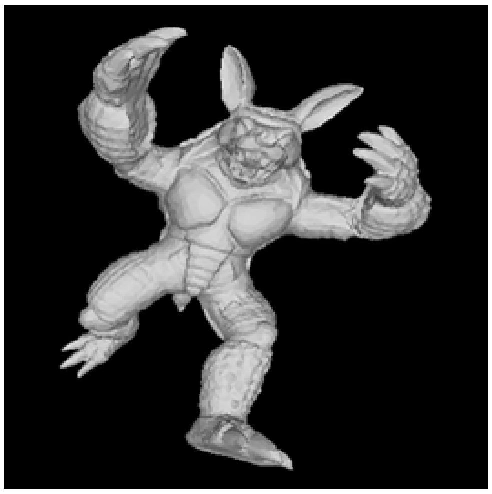
Results and Discussion

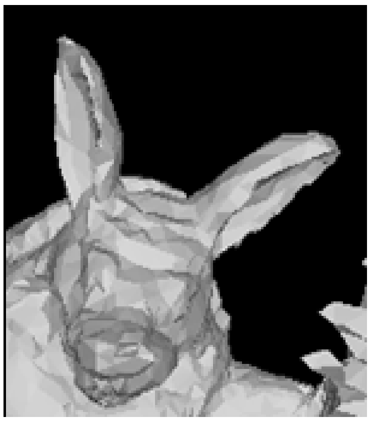
All experiments are performed on Intel core 2 duo E6750 2.66GHz PC with 3G ram, using NVIDIA Geforce 8600 GT graphics hardware and TAUCS [TCR03] version 2.2 linear solver.

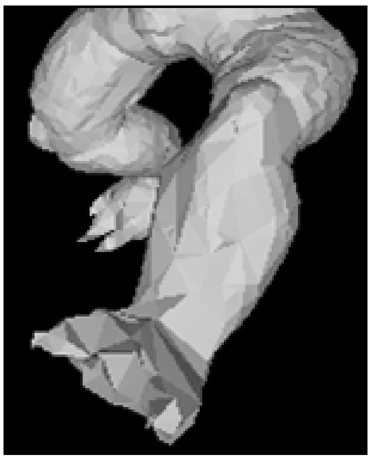
4.1 Detail preserving deformation results

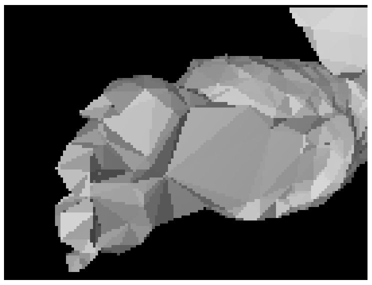
Figure 4.1, Figure 4.2 and Figure 4.3 show some results. Figure 4.1 and Figure 4.2 are results applied skeleton-based deformation on base mesh. Figure 4.3 is the result applied free-form deformation on base mesh.

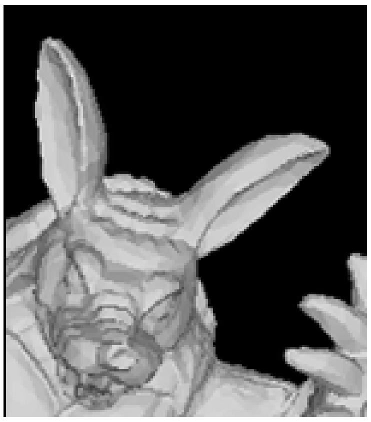

(a) Armadillo 10k model

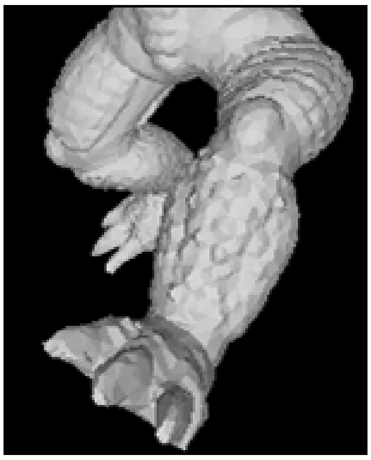

(b) Deformed Armadillo 10k model


(c) Deformed cut mesh shown on head


(d) Deformed cut mesh shown on leg


(e) Deformed cut mesh shown on hand


(f) Deformed mesh shown on head


(g) Deformed mesh shown on leg

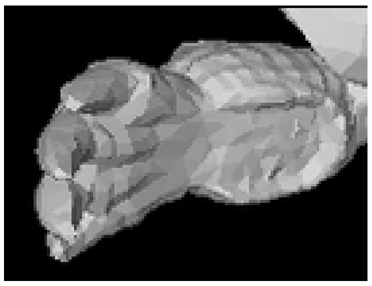
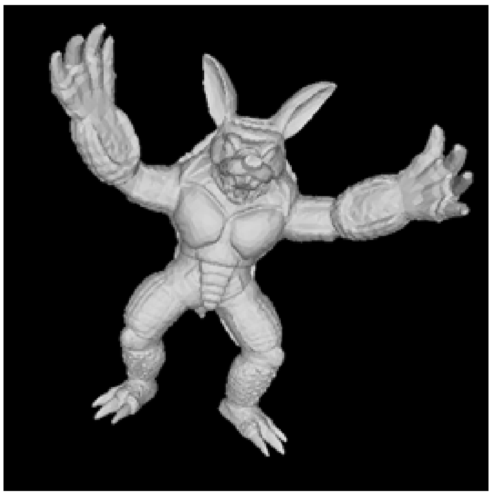
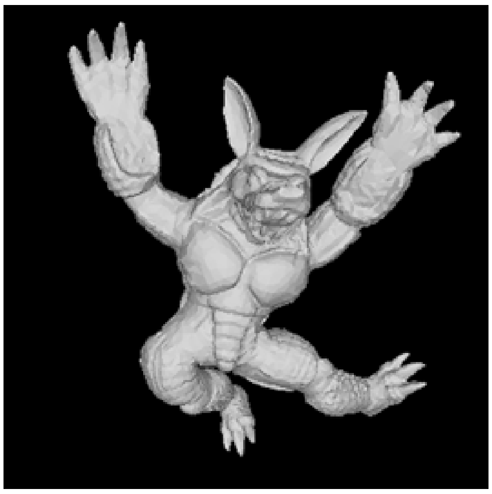
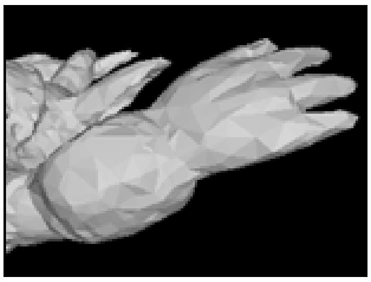
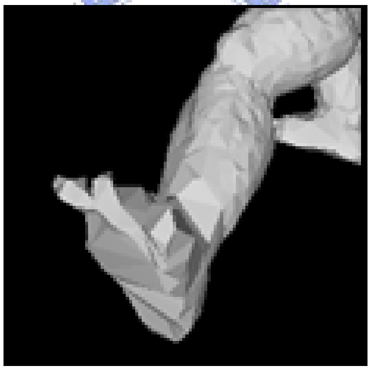

(h) Deformed mesh shown on hand

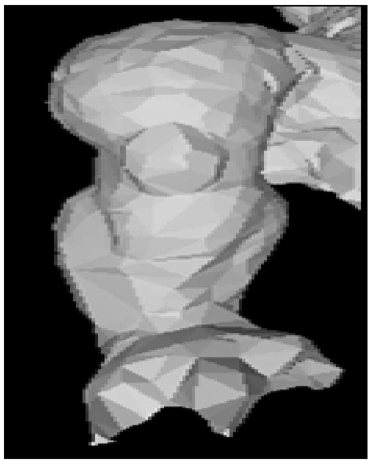
Figure 4.1: Armadillo 10k model deformation result.

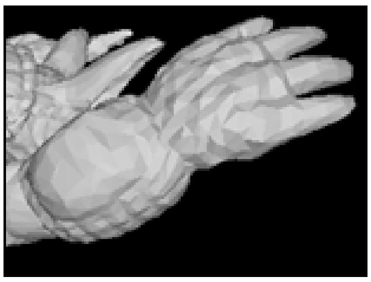

(a) Armadillo 10k model

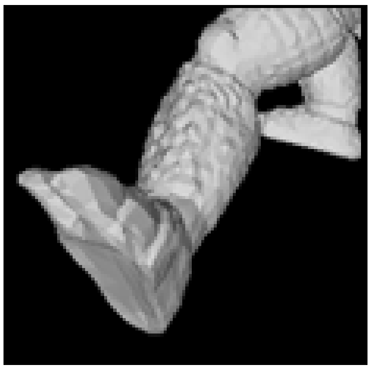

(b) Deformed Armadillo 10k model


(c) Deformed cut mesh shown on hand


(d) Deformed cut mesh shown on leg


(e) Deformed cut mesh shown on leg


(f) Deformed mesh shown on hand


(g) Deformed mesh shown on leg

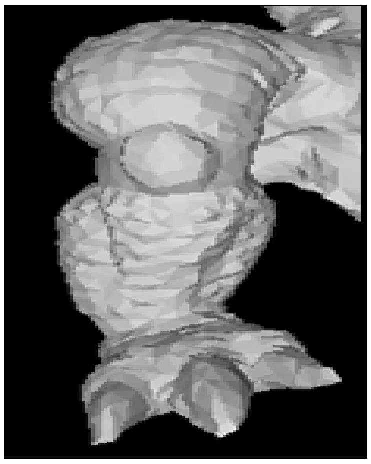
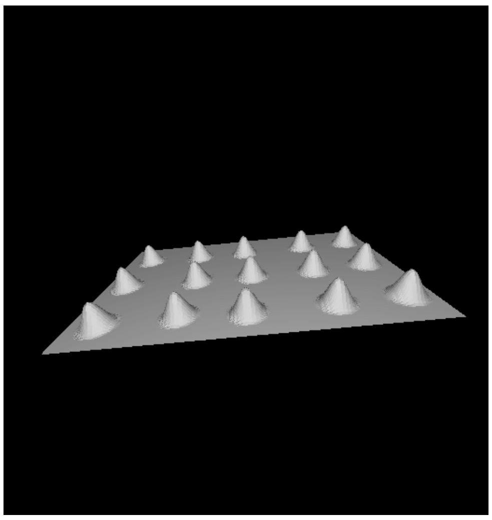
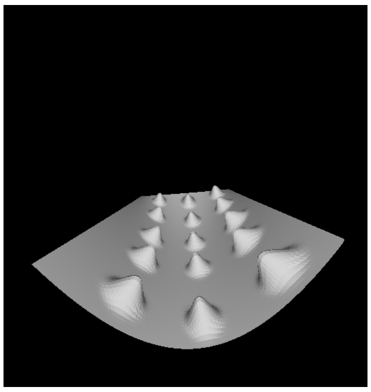
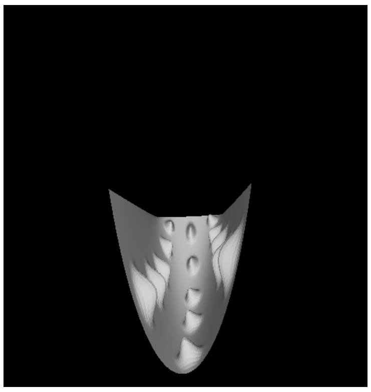
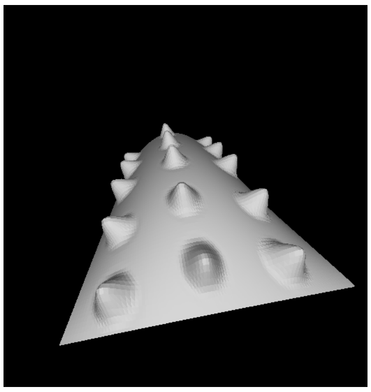

(h) Deformed mesh shown on leg

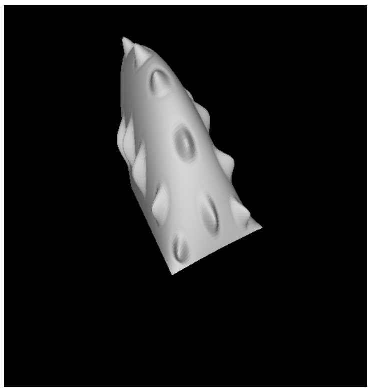
Figure 4.2: Armadillo 10k model deformation result.

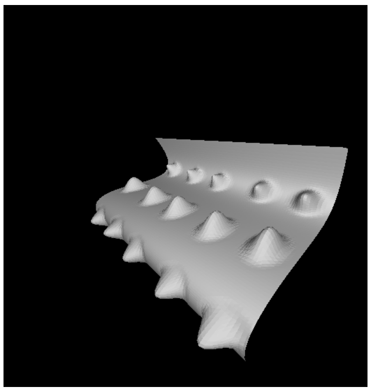

(a) 10k model


(b) deformation of model (a)


(c) deformation of model (a)


(d) deformation of model (a)


(e) deformation of model (a)


(f) deformation of model (a)

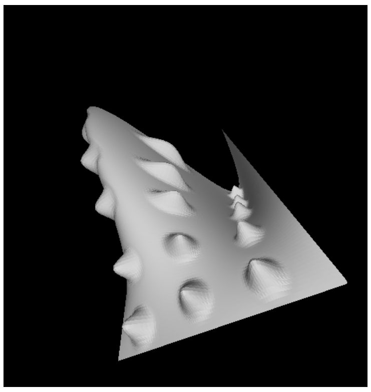

(g) deformation of model (a)

Figure 4.3: 10k model deformation results.

The detailed feature are smoothed on Figure 4.3. It is reasonable that under large deformation at feature bottom might stretch detail. Feature is reconstructed through optimization and will result in smoothed condition. Another example shows how detailed feature perform under smaller stretch is shown in Figure 4.4. Some constraints such as rigidly or volume preserving issue may be applied on optimization structure suit users' requirements.

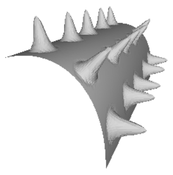
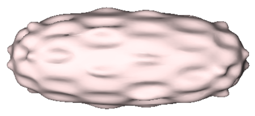
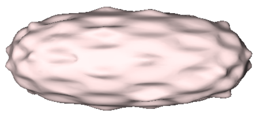


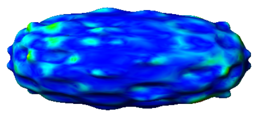
Figure 4.4: Feature is not smoothed under smaller stretch.

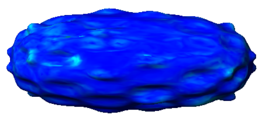
4.1.1 Performance and quality

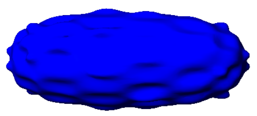
To compare performance and quality, we use a set level of detail models with different resolution in our experiments. Each model is applied the same deformation. Deformation results using whole mesh optimization is considered as ground truth. Deformation results applied our method are compared with whole mesh optimization results. We use metric on PolyMeCo [SMS05] such as geometric distance(comparison of two meshes by measuring local geometric differences between their surfaces), normal deviation(measurement of how much the normals of the vertices change from one model to another), Gaussian curvature deviation(comparison of mean and Gaussian curvature between two meshes), virtual metric [SCOT03] and mixed metric(weighting sum between geometric distance and normal deviation) to calculate difference between two deformed meshes(under the same deformation, applied our method or using whole mesh optimization). Examples are shown in Figure 4.5 and Figure 4.6.

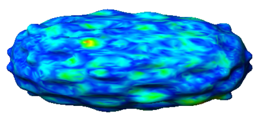

(a) Mesh deformation with whole optimization


(b) Mesh deformation with region optimization


(c) Geometric distance comparison


(d) Normal deviation comparison


(e) Gaussian curvature comparison


(f) Virtual metric comparison

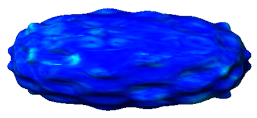
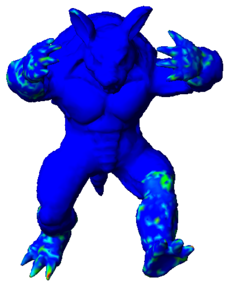
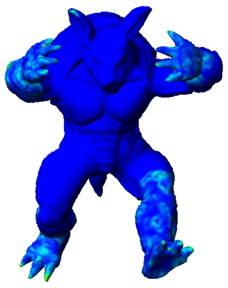

(g) Mixed measure comparison

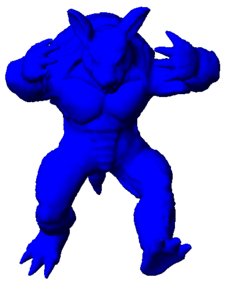
Figure 4.5: Comparison of 5k bumpy sphere deformation model using PolyMeCo.

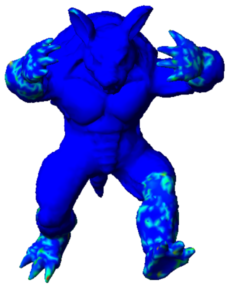
(a) Mesh deformation with whole optimization

(b) Mesh deformation with region optimization


(c) Geometric distance comparison


(d) Normal deviation comparison


(e) Gaussian curvature comparison


(f) Virtual metric comparison

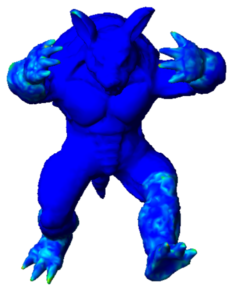

(g) Mixed measure comparison

Figure 4.6: Comparison of 10k Armadillo deformation model using PolyMeCo.

Table 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6 shows deformation results comparison(under same deformation, applied our method or using whole mesh optimization) on level of detail models.

	Performance	Threshold			
Type	200k	400k	600k	800k	1000k
whole optimization time(s)	4.549	4.549	4.549	4.549	4.549
region optimization time(s)	0.366	0.544	0.606	0.673	1.097
geometric distance	0.0388	0.03808	0.03728	0.03727	0.03617
normal deviation	0.0512	0.0489	0.04715	0.04714	0.04536
Gaussian curvature deviation	0.0309	0.0307	0.0306	0.0305	0.0301
virtual metric	8.4319	7.83361	7.4296	7.4295	7.1385
mixed metric	0.0512	0.0489	0.04715	0.04514	0.04536

Table 4.1: Performance and quality measurement on model with 2500 vertices.

From the experiments we found that with higher performance threshold, which means more vertices are applied region optimization rather than directly recover. It will have better quality(with fewer metric difference compared with whole mesh optimization) on every comparison metric supported by PolyMeCo but takes more time on region optimization. An example visualization on Table 4.2 is shown in 4.7.

A 2D example can be shown on 4.8.

4.2 Discussion

However, performance threshold only provides an upper bound of region optimization time. There is no strictly control on performance. As shown in Figure 4.9, there is no relationship on calculation time by applying the same performance threshold between different models. Because we only set it as an upper bound constraint. Region size distribution must be considered into cut updating criterion to have a more strict control on performance.

	Performance	Threshold			
Type	200k	400k	600k	800k	1000k
whole optimization time(s)	16.8486	16.8486	16.8486	16.8486	16.8486
region optimization time(s)	0.4054	0.52578	0.53444	0.80228	1.004
geometric distance	0.06978	0.06866	0.06865	0.06835	0.06783
normal deviation	0.0823	0.08144	0.08135	0.08086	0.08013
Gaussian curvature deviation	0.0251	0.02498	0.02498	0.0248	0.02473
virtual metric	12.128	12.0348	12.0261	11.9361	11.7673
mixed metric	0.0823	0.08144	0.0813	0.08086	0.08013

Table 4.2: Performance and quality measurement on model with 5000 vertices.

As explained above, after finishing updating cut, our deformed cut mesh is reconstructed through directly recover. We also have an experiment of optimizing cut mesh and then applying region optimization. Comparison between using directly recover or optimization applied on cut mesh is shown in Table 4.7. Figure 4.11 visualizes total simulation time and Figure 4.12 visualizes different cut mesh reconstruction approach comparisons(with whole optimization results).

Apply optimization on cut truly increase quality but sacrifice performance. According to the structure by optimizing cut mesh and apply region optimization below cut. Performance must take into consideration on balancing both stage. Lower cut means more region divided but cut mesh optimization takes more time. Higher cut means less cut mesh optimization time but fewer region divided.

From the experimental results we found that based on the structure of optimizing cut mesh and apply region optimization below cut, cut mesh vertices and region optimization vertices have similar quality. If we can also determine regions on cut mesh. Substituting whole optimization on cut mesh into region optimization. The result might have similar quality and increased performance.

An example of regions calculated on deformed mesh is shown in 4.10.

Type	Performance Threshold 200k	400k	600k	800k	1000k
whole optimization time(s)	47.2305	47.2305	47.2305	47.2305	47.2305
region optimization time(s)	0.6598	0.6651	0.9402	0.9752	1.1207
geometric distance	0.07169	0.07061	0.07022	0.07004	0.0696
normal deviation	0.08926	0.0885	0.08816	0.0879	0.0874
Gaussian curvature deviation	0.07782	0.0777	0.0776	0.0776	0.07739
virtual metric	14.4298	14.403	14.3866	14.3836	14.3421
mixed metric	0.08926	0.0885	0.08816	0.0879	0.0874

Table 4.3: Performance and quality measurement on model with 7500 vertices.

Type	Performance Threshold 200k	400k	600k	800k	1000k
whole optimization time(s)	74.0866	74.0866	74.0866	74.0866	74.0866
region optimization time(s)	0.4243	0.6506	0.9719	1.2249	1.3147
geometric distance	0.07975	0.07627	0.07215	0.07117	0.07085
normal deviation	0.09883	0.09618	0.09317	0.09251	0.09211
Gaussian curvature deviation	0.05789	0.057	0.0565	0.05587	0.0557
virtual metric	16.1379	16.0608	15.9716	15.9445	15.9348
mixed metric	0.09883	0.09618	0.09317	0.09251	0.09211

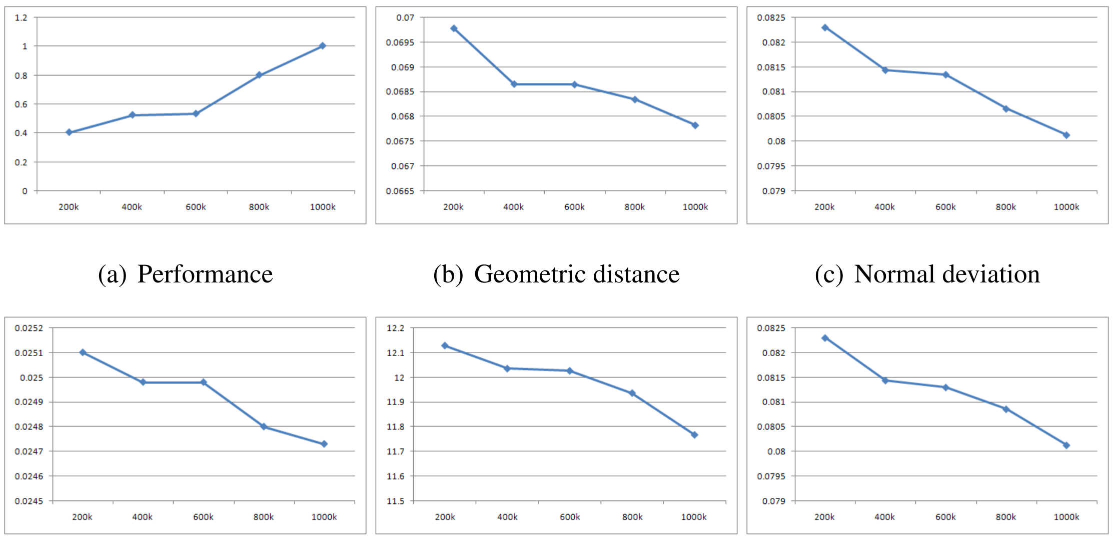
Table 4.4: Performance and quality measurement on model with 10000 vertices.

	Performance	Threshold			
Type	200k	400k	600k	800k	1000k
whole optimization time(s)	113.716	113.716	113.716	113.716	113.716
region optimization time(s)	0.3099	0.5805	0.79	0.851	1.03
geometric distance	0.0736	0.06959	0.06799	0.06735	0.0673
normal deviation	0.093	0.0901	0.0888	0.0883	0.0881
Gaussian curvature deviation	0.0385	0.03743	0.03715	0.0371	0.037
virtual metric	16.2099	16.1043	15.9572	15.8603	15.85
mixed metric	0.093	0.0901	0.0888	0.0883	0.0881

Table 4.5: Performance and quality measurement on model with 12500 vertices.

	Performance	Threshold			
Type	200k	400k	600k	800k	1000k
whole optimization time(s)	166.561	166.561	166.561	166.561	166.561
region optimization time(s)	0.6619	0.8576	1.0818	1.2608	1.352
geometric distance	0.06418	0.06292	0.06215	0.06166	0.06152
normal deviation	0.08996	0.0887	0.08807	0.0876	0.08751
Gaussian curvature deviation	0.05484	0.05438	0.05388	0.05287	0.05282
virtual metric	14.8395	14.8156	14.7985	14.7931	14.7886
mixed metric	0.08996	0.0887	0.08807	0.0876	0.08751

Table 4.6: Performance and quality measurement on model with 15000 vertices.



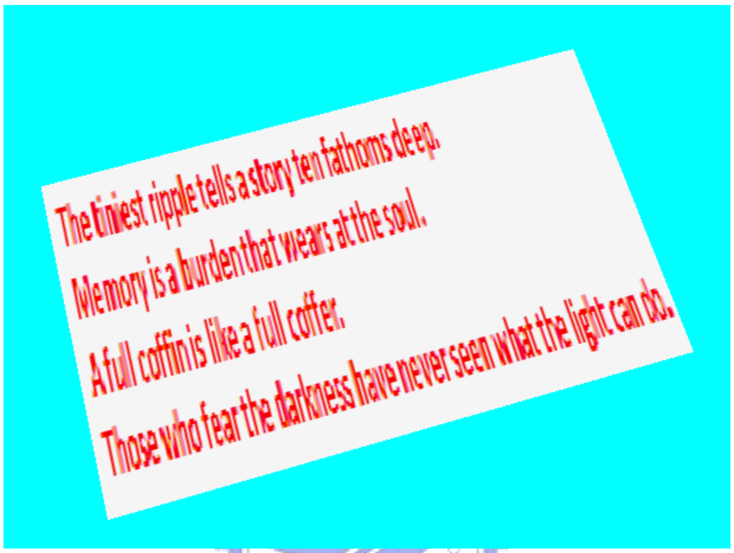
(a) Performance (b) Geometric distance (c) Normal deviation

(d) Gaussian curvature deviation (e) Virtual metric (f) Mixed metric

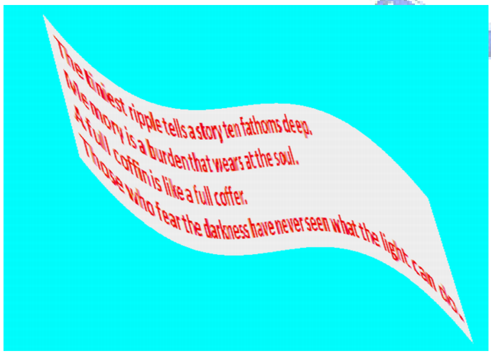
Figure 4.7: Visualization of experiment on model with 5000 vertices.

Type	directly recover	cut optimization
Total time(s)	1.0678	5.0058
geometric distance	0.0678	0.0401
normal deviation	0.0801	0.056
Gaussian curvature deviation	0.0247	0.0167
virtual metric	11.7073	9.208
mixed metric	0.0801	0.056

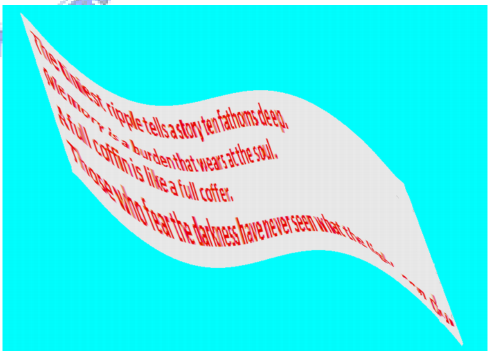
Table 4.7: Performance and quality measurement on model with 5000 vertices with different approach on cut mesh. 2000 vertices are directly recovered(or optimized) and 2000 vertices are reconstructed through region optimization.


The tiniest ripple tells a story ten fathoms deep.
Memory is a burden that wears at the soul.
A full coffin is like a full coffer.
Those who fear the darkness have never seen what the light can do.

(a) Original mesh.


The tiniest ripple tells a story ten fathoms deep.
Memory is a burden that wears at the soul.
A full coffin is like a full coffer.
Those who fear the darkness have never seen what the light can do.

(b) Deformation with detail-preserving



(c) Deformation without detail-preserving

Figure 4.8: Deformation example on 2D.

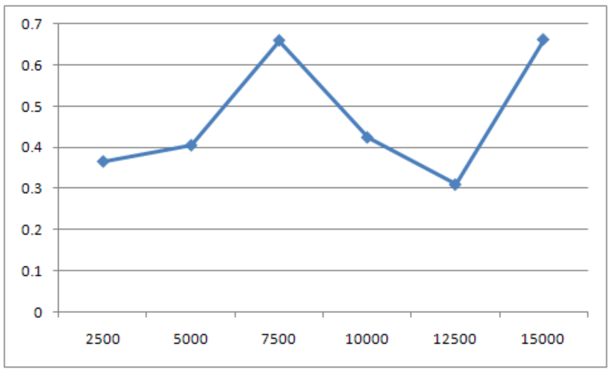
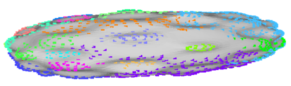
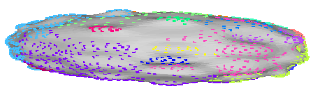


Figure 4.9: Comparison of the region optimization time with the same threshold between different models.


(a) Region example.


(b) Region example.

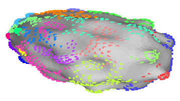

(c) Region example.

Figure 4.10: Region visualization.

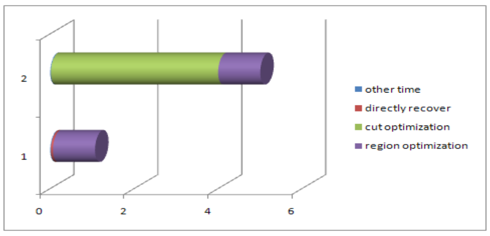
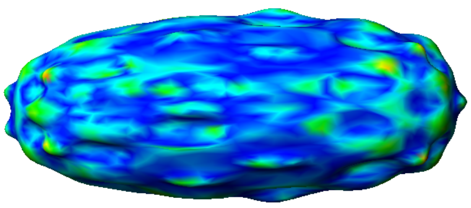
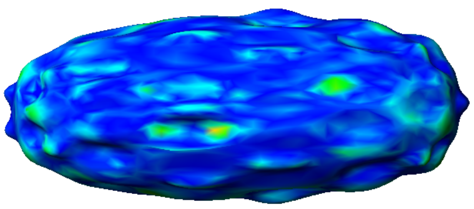
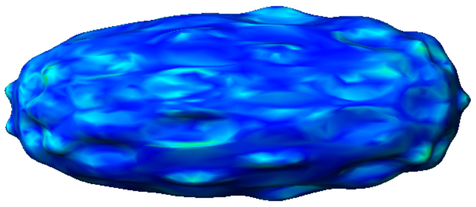
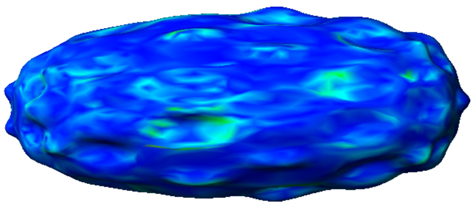


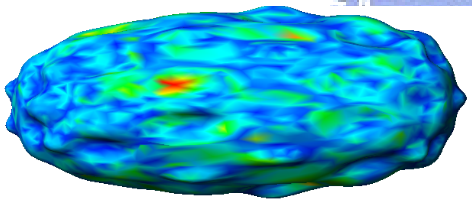
Figure 4.11: Simulation time Comparison between directly recover(1) or optimization(2) applied on cut mesh.

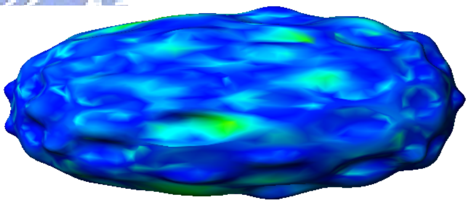

(a) Geometric distance.

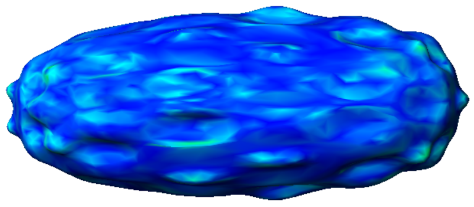

(b) Geometric distance.


(c) Normal deviation.


(d) Normal deviation.


(e) Virtual metric.


(f) Virtual metric.


(g) Mixed metric.

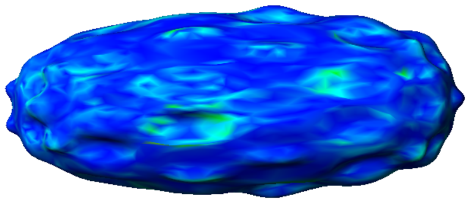

(h) Mixed metric.

Figure 4.12: Comparison between directly recover(left) or optimization(right) applied on cut mesh.

An experiment of effect on different cut updating criterion is shown in Figure 4.13. Corresponding region information is shown in 4.14. We found that considering normal difference to maintain region boundary smoothness determines region better. However, this term is hard to catch acute deformation for the reason that region with acute deformation might easily results in larger normal difference in corresponding edges. So geometric stretch and normal difference must be considered together to suit our needs: catching acute deformation and maintaining smooth region boundary.

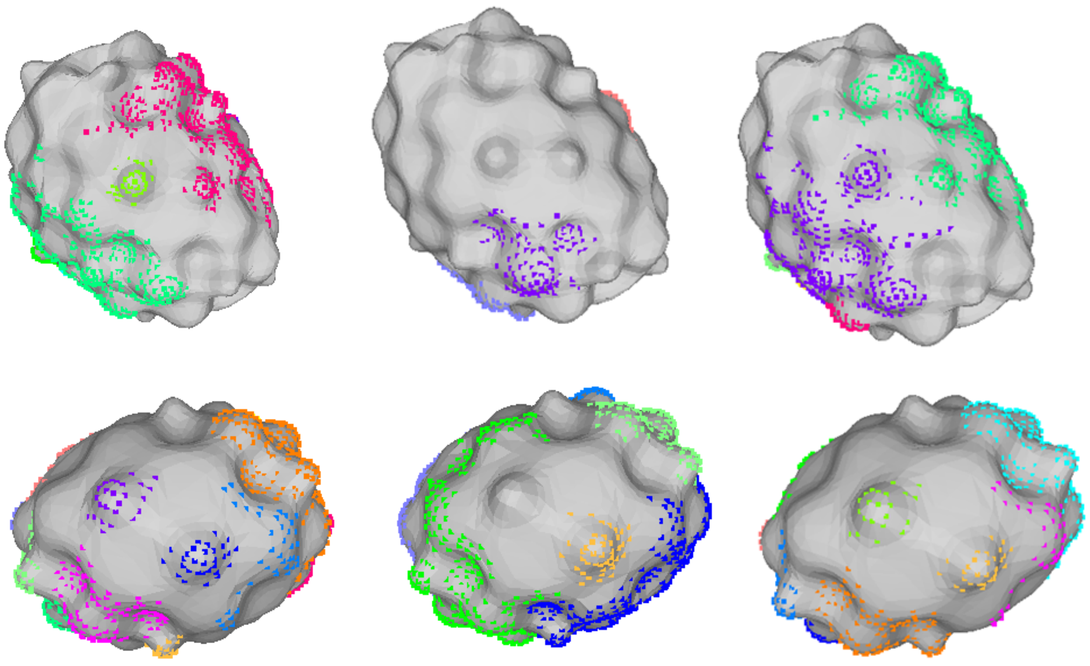

(a) Cut update criterion considers geometric stretch only. (b) Cut update criterion considers normal difference only. (c) Cut update criterion considers both geometric stretch and normal difference.

Figure 4.13: Different cut update criterion.

The hierarchical structure plays an important role on optimized region determination. As explained in 3.1.1, a hierarchical dependency structure is built and nodes representing removed vertices are merged during hierarchy construction according to neighborhood relationship. The hierarchical structure is limited on losing relationship between coarse mesh vertices and re-

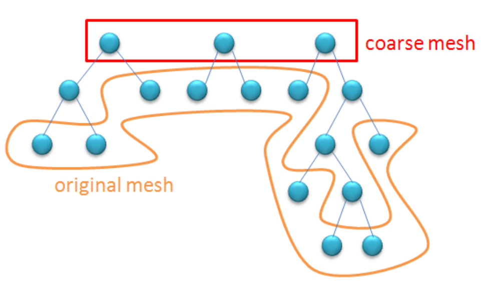
geometric stretch	normal difference	geometric stretch + normal difference
21	102	21
27	30	58
23	34	23
34	145	34
25	221	25
23	41	24
30	224	30
58	324	59
25		24
42		42
225		240
98		125
24		34
25		22
439		419
367		283

Figure 4.14: Optimized region vertex number on different cut updating criterion as shown in Figure 4.13.

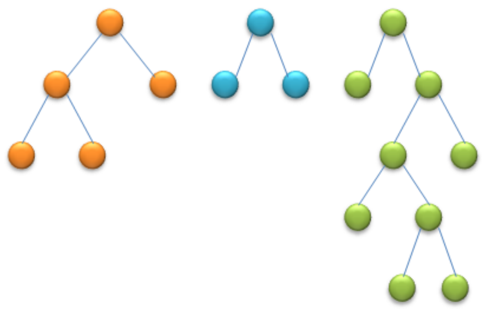
moved vertices(nodes in hierarchy) because hierarchy only contains removed vertices.

We have an experiment on different hierarchical structure approach. We choose vertex hierarchy structure similar to view-dependent LOD [Hop97]. However, apply the hierarchical structure in our algorithm is limited on half-edge collapse simplification operator. It is because in our algorithm, regions are determined by sub-tree. So merged vertex must connect all one ring neighborhood vertices of removed vertex after triangulating naked area to avoid vertex disconnection on region. An example of using vertex removal operation and vertex disconnection happens on region is shown in Figure 4.17. The problem results in new connectivity happens on new generated triangles by vertex removal. To avoid this problem on vertex removal operation, new generated connectivity must be considered into hierarchy building process.

When we choose half-edge collapse as simplification operator, after building hierarchy there results regions with the same number of coarse mesh vertices. In conventional vertex hierarchy, root of the hierarchy means coarse mesh and the leaves of the hierarchy means original mesh. Each sub-tree represents an independent region as shown in Figure 4.15. However, the hierarchy results in lots of tiny regions(the same number as coarse mesh vertex numbers) and doesn't



(a) Vertex hierarchy.



(b) Sub-tree represents region in vertex hierarchy.

Figure 4.15: Vertex hierarchy and region.

suit our needs. Merging region with dependency criterion then must be considered into cut update operation. When using vertex removal operation, the hierarchy must be considered new generated vertex connectivity into parent-child relationship to avoid vertex disconnection on region. As shown in Figure 4.16.

The reason that conventional vertex hierarchy [Hop97, XV96] doesn't suit our objective is because our region requirement. Topology connectivity in region must be considered into hierarchy building process and incident cut update criterion to merge or subdivide region. Dependency is checked according to edge dependency relationship as explained on Section 3.1.1.

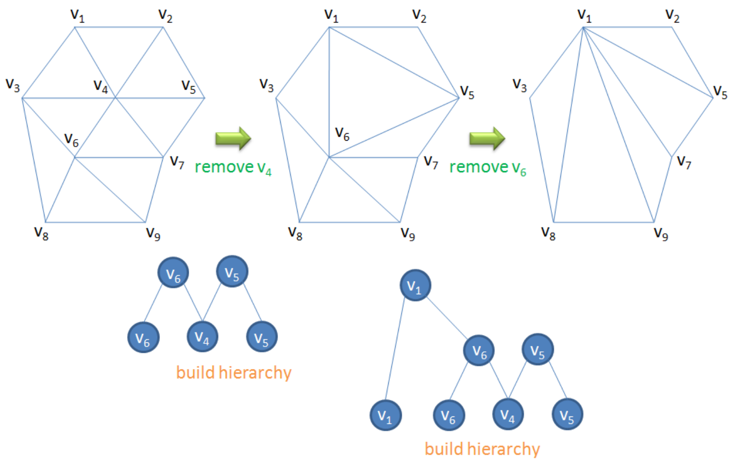
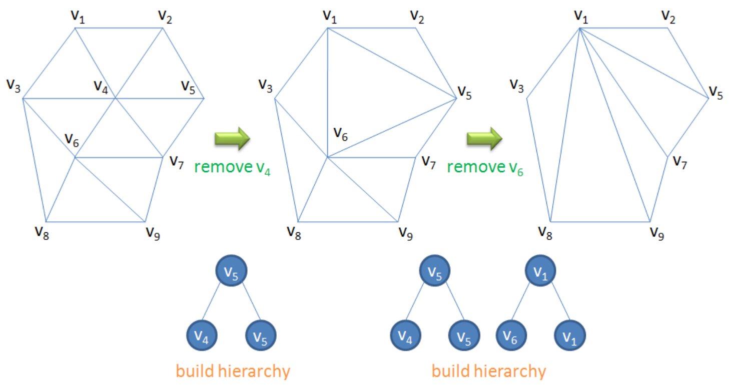
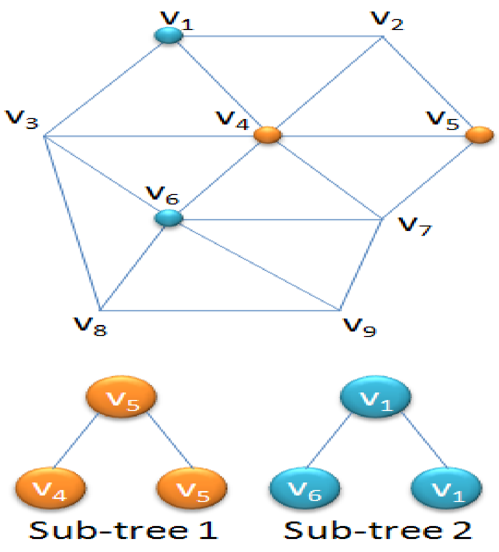


Figure 4.16: Build vertex hierarchy upon vertex removal.



(a) Build vertex hierarchy.



(b) Disconnection happens on region.

Figure 4.17: Vertex disconnection is happened on sub-tree region when using vertex removal operation.

CHAPTER 5

Conclusion

In this thesis, We proposed a detail-preserving deformation method with balanced performance and quality during dynamic simulation. The key idea is building hierarchy to increase performance without losing detail control. Finding cut in hierarchy to subdivide mesh into regions and optimizing them to get original mesh deformation.

Our hierarchy is built for convenient handling vertex removal/recover operation dependency problems. Dependency is implied on parent and child relationship. To check a valid vertex removal/recover operation, we only need to check if its child/parent are removed/recovered already.

However, a node in our hierarchy means a vertex removal/recover operator. All nodes in hierarchy only contains vertices between original mesh and base mesh(vertices simplified at pre-processing stage). Base mesh vertices are considered as boundary constraint during region optimization. Different deformation causes different regions to optimize. Some base mesh vertices might not join region optimization and their positions are kept. These vertices don't have chance to update their position once base mesh deformed. Since we hope that all vertices might have chance to update their position. A simplest way is that we optimize base mesh first after it deforms. It takes about one second on base mesh resolution between 1000 to 1500 vertices. Another complicate way is to update the hierarchy in order to contain base mesh vertices. We have an experiment on Section 4.2, we have discussed how different hierarchical structure affects region specification. Find a more suitable hierarchy and corresponding cut updating criterion will be the future work.

Bibliography

[BK03] Mario Botsch and Leif Kobbelt. Multiresolution surface representation based on displacement volumes. *Comput.Graph.Forum*, 22(3):483–492, 2003. Available from World Wide Web: `http://dblp.uni-trier.de/db/journals/cgf/cgf22.html#BotschK03`.

[BWHT07] Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. A finite element method for animating large viscoplastic flow. *ACM Trans.Graph.*, 26(3), 2007.

[CBP05] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. Particle-based viscoelastic fluid simulation. In *Symposium on Computer Animation 2005*, pages 219–228, july 2005.

[CGC$^+$02a] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovi. Interactive skeleton-driven dynamic deformations, 2002. Available from World Wide Web: `citeseer.ist.psu.edu/capell02interactive.html`.

[CGC$^+$02b] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popović. A multiresolution framework for dynamic deformations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 41–47, New York, NY, USA, 2002. ACM.

[CHP89] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 243–252, New York, NY, USA, 1989. ACM. Available from World Wide Web: `http://doi.acm.org/10.1145/74333.74358`.

[CMO97] Jonathan Cohen, Dinesh Manocha, and Marc Olano. Simplifying polygonal models using successive mappings. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 395–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[CZ92] David T. Chen and David Zeltzer. Pump it up: computer animation of a biomechanically based model of muscle using the finite element method. *SIGGRAPH Comput.Graph.*, 26(2):89–98, 1992. Available from World Wide Web: `http://doi.acm.org/10.1145/142920.134016`.

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space and time adaptive sampling. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 31–36. ACM Press / ACM SIGGRAPH, 2001. Available from World Wide Web: `citeseer.ist.psu.edu/debunne01dynamic.html`.

[DpG95] Mathieu Desbrun and Marie paule Gascuel. Animating soft substances with implicit surfaces. In *In SIGGRAPH 95 Conference Proceedings, Annual Conference Series*, pages 287–290. ACM SIGGRAPH, Addison Wesley, 1995.

[GTT89] JeanPaul Gourret, Nadia Magnenat Thalmann, and Daniel Thalmann. Simulation of object and human skin deformations in a grasping task. *Computer Graphics*, 23(3):21–30, 1989. Available from World Wide Web: `citeseer.ist.psu.edu/gourret89simulation.html`.

[Hop96] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 99–108, New York, NY, USA, 1996. ACM.

[Hop97] Hugues Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 189–198, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[HSL$^+$06] Jin Huang, Xiaohan Shi, Xinguo Liu, Kun Zhou, LiYi Wei, ShangHua Teng, Hujun Bao, Baining Guo, and HeungYeung Shum. Subspace gradient domain mesh deformation. 25(3):1126–1134, 2006. Available from World Wide Web: `http://doi.acm.org/10.1145/1141911.1142003`. ACM Trans.Graph.

[ITA04] G. IRVING, J. TERAN, and R. AND. Invertible finite elements for robust simulation of large deformation, 2004. Available from World Wide Web: `citeseer.ist.psu.edu/irving04invertible.html`.

[KCVS98] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and HansPeter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114, New York, NY, USA, 1998. ACM. Available from World Wide Web: `http://doi.acm.org/10.1145/280814.280831`.

[LLCO08] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Trans. Graph.*, 27(3):1–10, 2008.

[LMH00] Aaron Lee, Henry Moreton, and Hugues Hoppe. Displaced subdivision surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[LS08] Torsten Langer and Hans-Peter Seidel. Higher order barycentric coordinates. In George Drettakis and Roberto Scopigno, editors, *Eurographics 2008*, volume 27, pages 459–466, Crete, Greece, 2008. The European Association for Computer Graphics (Eurographics), Foundation for Research and Technology - Hellas (FORTH), Blackwell.

[LSC+04] Yaron Lipman, Olga Sorkine, Daniel CohenOr, David Levin, Christian Rossl, and HansPeter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.

[LSLC05] Yaron Lipman, Olga Sorkine, David Levin, and Daniel CohenOr. Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH 2005*, pages 479–487. ACM Press, 2005.

[LSS+98] A. W. F. Lee, W. Sweldens, P. Schrder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. *Computer Graphics Proceedings (SIGGRAPH 98)*, pages 95–104, 1998.

[LVJ05] Chang Ha Lee, Amitabh Varshney, and David W. Jacobs. Mesh saliency. *ACM Trans.Graph.*, 24(3):659–666, 2005. Available from World Wide Web: `http://doi.acm.org/10.1145/1073204.1073244`.

[MBK07] Martin Marinov, Mario Botsch, and Leif Kobbelt. Gpu-based multiresolution deformation using approximate normal field reconstruction. *journal of graphics tools*, 12(1):27–46, 2007,.

[MDM+02] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable real-time deformations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 49–54, New York, NY, USA, 2002. ACM.

[MDSB02] M. MEYER, M. DESBRUN, P. SCHR, and A. BARR. Discrete differentialgeometry operators for triangulated 2-manifolds, 2002. Available from World Wide Web: `citeseer.ist.psu.edu/meyer02discrete.html`.

[MG04] Matthias Müller and Markus Gross. Interactive virtual materials. In *GI '04: Proceedings of Graphics Interface 2004*, pages 239–246, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[MKN$^+$04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 141–151, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[MMDJ01] Matthias Müller, Leonard McMillan, Julie Dorsey, and Robert Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, pages 113–124, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[MTG04] Matthias Muller, Matthias Teschner, and Markus Gross. Physically-based simulation of objects represented by surface meshes. In *CGI '04: Proceedings of the Computer Graphics International*, pages 26–33, Washington, DC, USA, 2004. IEEE Computer Society.

[NISA06] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389, New York, NY, USA, 2006. ACM. Available from World Wide Web: `http://doi.acm.org/10.1145/1174429.1174494`.

[OBH02] James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph.*, 21(3):291–294, 2002.

[OH99] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 137–146, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[PB81] Stephen M. Platt and Norman I. Badler. Animating facial expressions. *SIGGRAPH Comput.Graph.*, 15(3):245–252, 1981. Available from World Wide Web: `http://doi.acm.org/10.1145/965161.806812`.

[PDA01] G. Picinbono, H. Delingette, and N. Ayache. Non-linear and anisotropic elastic soft tissue models for medical simulation, 2001. Available from World Wide Web: `citeseer.ist.psu.edu/picinbono01nonlinear.html`.

[SCO04] Olga Sorkine and Daniel Cohen-Or. Least-squares meshes. In *Proceedings of Shape Modeling International*, pages 191–199. IEEE Computer Society Press, 2004.

[SCOT03] Olga Sorkine, Daniel Cohen-Or, and Sivan Toledo. High-pass quantization for mesh encoding. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 42–51. Eurographics Association, 2003.

[SLC$^+$04] Olga Sorkine, Yaron Lipman, Daniel CohenOr, Marc Alexa, Christian Rossl, and HansPeter Seidel. Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. Eurographics Association, 2004.

[SMS05] Samuel Silva, Joaquim Madeira, and Beatriz Sousa Santos. Polymeco: A polygonal mesh comparison tool. *Information Visualisation, International Conference on*, 0:842–847, 2005.

[SSP07] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. 26(3):80, 2007. Available from World Wide Web: `http://doi.acm.org/10.1145/1276377.1276478`. ACM Trans.Graph.

[TBHF03] J. TERAN, S. BLEMKER, V. HING, and R. FED. Finite volume methods for the simulation of skeletal muscle, 2003. Available from World Wide Web: `citeseer.ist.psu.edu/teran03finite.html`.

[TCR03] Sivan Toledo, Doron Chen, and Vladimir Rotkin. Taucs: A library of sparse linear solvers, version 2.2. 2003. Available from World Wide Web: `http://www.tau.ac.il/~stoledo/taucs/`.

[TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 269–278, New York, NY, USA, 1988. ACM.

[TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM.

[TW88] Demetri Terzopoulos and Andrew Witkin. Physically based models with rigid and deformable components. *IEEE Comput.Graph.Appl.*, 8(6):41–51, 1988.

[Wat87] Keith Waters. A muscle model for animation three-dimensional facial expression. *SIGGRAPH Comput.Graph.*, 21(4):17–24, 1987. Available from World Wide Web: `http://doi.acm.org/10.1145/37402.37405`.

[WT91] K. Waters and D. Terzopoulos. Modeling and animating faces using scanned data. *JVCA*, 2:123–128, 1991.

[XV96] Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 327–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[ZHS+05] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and HeungYeung Shum. Large mesh deformation using the volumetric graph laplacian. 24(3):496–503, 2005. Available from World Wide Web: `http://doi.acm.org/10.1145/1073204.1073219`. ACM Trans.Graph.