Ph.D Thesis

Research of Spherical Style Deformation on Single Component Models

Feng Xuemei

Department of Design and Media Technology
Graduate School of Science and Engineering
Iwate University

September 2023

Abstract

We present a spherical style deformation algorithm on single component models that can deform the models with spherical style, while preserving local details of the original models. Because 3D models have complex skeleton structures that consist of many components, the deformation around connections between each single component are complicated, especially preventing mesh self-intersections. To the best of our knowledge, we could not find not only methods to achieve a spherical style in a 3D model consisting of multiple components but also methods of a single component. In this thesis, we focus on spherical style deformation of single component models, and propose a method which deforms the input model with the spherical style, while preserving the local details of the input model. We explore a cluster of linear features of the sphere shape and describe these features as ℓ_2-regularization. According to the feature descriptions, energy function is established which combines the ARAP term and the spherical term. An efficient optimization solution is also provided to solve the energy function. We have performed our method on convex and smooth models, convex and sharp models, finally complex models with different linear spherical features respectively. In our experiments, energy can be well converged. Based on these experimental results, we analyze the effect of each feature on spherical style deformation for single component models and achieve a most suitable feature for deformation. Our approach can deform the input model smooth, rounder and curved successfully, while preserving local details of the original input model. At the same time, we showed deformation results of different sphere center positions. We also compared our experimental results with the 3D geometric stylization method of normal-driven spherical shape analogies and confirmed that our method successfully deforms models smooth, rounder, and curved. Limitations, problems and future work of our method based on the experimental

results are also discussed.

We also found that the results of our deformation are dependent on the quality of the input mesh. When the input mesh consists of many obtuse triangles, it leads to potential oscillation of the numerical method, poorly conditioned matrices, worsening the speed and accuracy of the linear solver and above spherical style deformation method fails. To solve this problem, we propose an optional deformation method based on convex hull proxy model as the complementary deformation method. Our proxy method constructs the proxy model of the input model and applies above spherical style deformation method to the proxy model. Finally, deformation result of the input model is obtained by the projection calculation between the proxy model and the input model and interpolation method between the input model and the deformed proxy model. We performed this proxy method to the obtuse triangle mesh and confirmed that the method can achieve better results, such as smoother surface, compared with above spherical style deformation method. At the same time, various t functions and partial deformation options give more kinds of deformation possibilities. Finally, we discuss the limitations of the proxy method.

Acknowledgements

Thanks my supervisor Prof.Katsutsugu Matsuyama. Thanks him for accepting me and giving me the opportunity to do a PhD. With the help of his educational philosophy, I realized my research interests. Thanks him for giving me great freedom in research. Under the guidance of my supervisor, I was able to complete research topic selection, experiments, thesis and so on. I was impressed by his very serious attitude in revising my paper and the attention to details. Because of not really understanding the details, I made funny mistakes that I will never forget. Finally, I would like to thank Matsuyama sensei for the suporting of computer equipments and books resources.

Thanks Prof.Kouichi Konno. I remember that when I first communicated and discussed about one paper with Konno sensei, he first picked up the formulas in paper to read. The habit of reading formulas first, at that moment, reminded me of the situation I communicated with Prof.Zhiyi Zhang. Prof.Zhang also has the habit of reading formulas first. Although Iwate University and Northwest A&F University are far apart, the same academic style made my communication with Prof.Konno very cordial. In the following research time, I would like to thank Konno sensei for his selfless devotion, enthusiastic help and encouragement. The time spent with Konno sensei is unforgettable. After finishing asking Konno sensei questions, almost all were left, the whole corridor was dark, and only Konno sensei's lab was lit. Next day morning, received relevant research papers shared by Konno sensei. The mailbox received the revision comments sent by Konno sensei at 1:00 in the morning. When I met Prof.Konno, he enthusiastically asked me about my current research situation. When I encountered problems and was confused, I knocked on the door of Prof.Konno's office, and I was grateful to be able to communicate with Konno sensei in time. Finally, I am very grateful that he

can spare energy from his busy work to help me. Konno sensei's enthusiasm and passion for research deeply affect me. Thanks for his help.

Thanks Prof.Zhang, although we are far apart, when I encounter difficulties, he is always very enthusiastic to help me. Thanks Dr.Qing Fang. Although we never met, he was very sincere, communicating with me about my research process. His dedication to research has set a good example for me. Thanks Dr.HsuehTi Derek Liu. When I asked questions about his paper several times by email, he always answered me quickly and friendly. Thanks to Xizhen Xiao. Thank you for his patience and help.

Last but not the least, thanks Prof.Mitsugu Saito and Prof.Tadahiro Fujimoto who discussed in revising my thesis.

Thanks my father. Whether happiness or sadness, my father sets me a good example to face them calmly. Thanks him for accompanying me growing up like a friend.

Finally, thanks Iwate international communication association for helping me financially.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Stylization of 3D Shape Models

Style transfer is a hot research topic in computer graphics. With the development of artificial intelligence, style transfer has achieved amazing results especially in the field of 2D images. Images can be stylized by changing parameters, which is a very simple way [17]. However, 3D geometric style transfer is still a challenging research topic, due to the complexity of geometries.

Stylization of 3D models refers to the techniques of deforming or changing the input model with desired style. We divide 3D style transfer of geometries into two categories: local style transfer, and global style transfer.

1.1.1 Local Style Transfer

The first category of 3D geometric style transfer is local style transfer. Local style transfer clones local details of the style shape onto the input shape and mapping between the style shape and the input shape is known. For example, Takayama K and Schmidt [21] have developed a simple brush interface to achieve local geometric style transfer interactively, shown as Fig.1.1. Recently, Chen Z and Kim V G [4] adopt a fully automatic way to realize local geometric style transfer, shown as Fig.1.2 and 1.3.

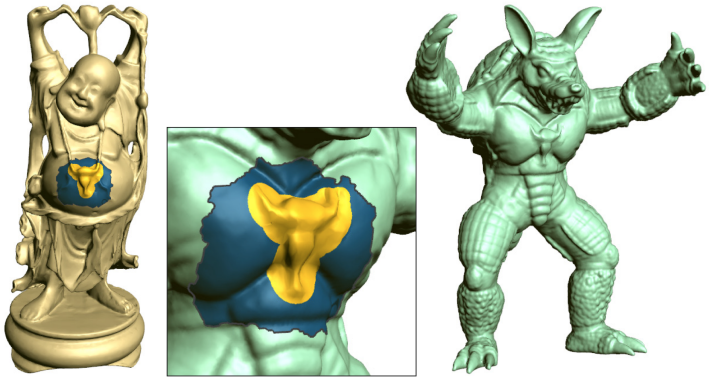


Figure 1.1: GeoBrush enables to clone various areas (gold) from a selected canvas region onto a target surface [21]. Target surface is the cyan-blue surface.

1.1.2 Global Style Transfer

The second category is global style transfer, which is global shape-related. The core of global style transfer is the establishment of the mapping relationship between the input model (or a base model) and the style model (or style operator). We summarize the global style transfer methods into two categories. One is the skeleton-based approach, and another one is normal-based approach.

1.1.2.1 Skeleton-based Approach

Skeleton-based approach obtains point-to-point mapping through skeleton information and mesh parameterization methods. Firstly, extract skeleton information of the two input models (a base model and a style model) and obtain coarse mapping of the two input models through skeleton and some design constraints. Then, based on coarse mapping of the two input models, achieve point-to-point mapping through mesh parameterization. Finally, according to the mapping relationship, deform one input model (the base model) with the style of another input model [6, 7].

In order to achieve coarse mapping of the two input models, the method requires complex preprocessing, such as shape segmentation, shape graph construction, correspondence search. Duncnan et al.[6] (shown in Fig.1.4) and

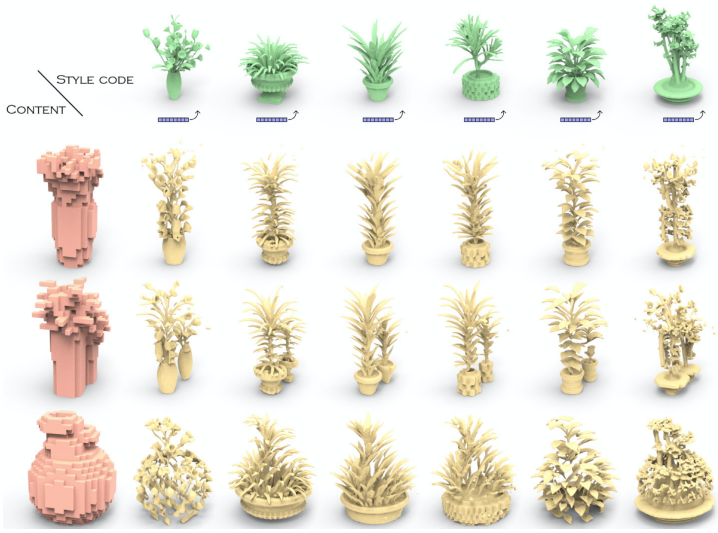


Figure 1.2: The 3D detailization network, DECOR-GAN, refines a coarse shape (red, leftmost) into a variety of detailed shapes, each conditioned on a style code characterizing an exemplar detailed 3D shape (green, topmost) [4].

Huang et al.[7] (shown in Fig.1.5) take two models as input, a base model and a style model, and generate a new model by mesh surgery (segmenting or shape graph construction) and topology blending of two models to get coarse mapping correspondence, and merging the two models according to mapping correspondence relationship. It is from differential view to achieve final point-to-point mapping. The approach can deform the base model with complex styles. However, this kind of methods achieve mapping correspondence through complex operations, especially coarse mapping achievement.

1.1.2.2 Normal-based Approach

Normal-based approach is not from differential geometry view. The mapping correspondence is different from the point-to-point mapping correspondence through mesh parameterization method. So far, the existing research establishes mapping correspondence through the normal vector distribution. The base model (or the input model) has approximately same vector distribution as the style model.

The normal-based approach [13, 14, 11] deforms the input model by chang-

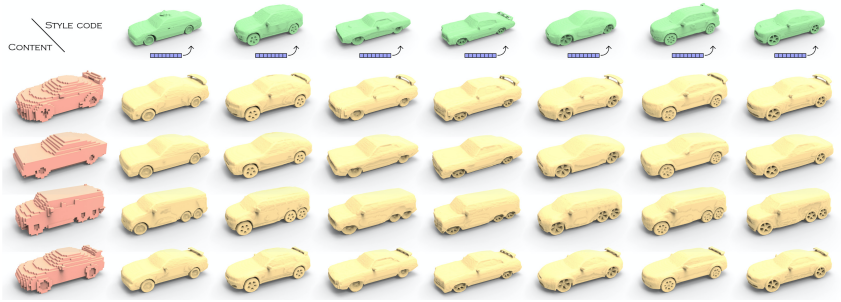


Figure 1.3: Results by upsampling coarse voxels with different style codes [4]. The detailed shapes are shown on the top that correspond to the input style codes. The input coarse voxel models are shown on the lest.

ing the distribution of surface normals so that the surface directs towards the desired direction, shown as Fig.1.6, 1.7, and 1.8. In Fig.1.6, the style model is cube, and style the input model (bunny model) with cubic style. In Fig.1.7, style models are represented by green models, and style the input model (the cow model) with correspondence style models. In Fig.1.8, the style feature is defined by liner combination of Gaussian functions. In every group result, the top left model is the combination of Gaussian functions. Style the input model (the bottom left model) with correspondence style feature, and orange model is styled result.

The normal-based approach has the advantage that no mesh surgery is required, and the topology of the deformed model is not changed, making it easier to understand the features of the input model in situations such as post-processing. This kind of methods style 3D models in a simple way, such as adjusting parameters, without complex operations. As far as our survey, studies under this approach employ polyhedra as their style, deforming surfaces into various polygon planes. For example, Liu and Jacobson [13] achieve cubic stylization by targeting surfaces parallel to xyz-axes in the object coordinate system. They perform optimization to generate stylized models, while preserving the details of the input model with As-Rigid-As-Possible(ARAP) [20] method.

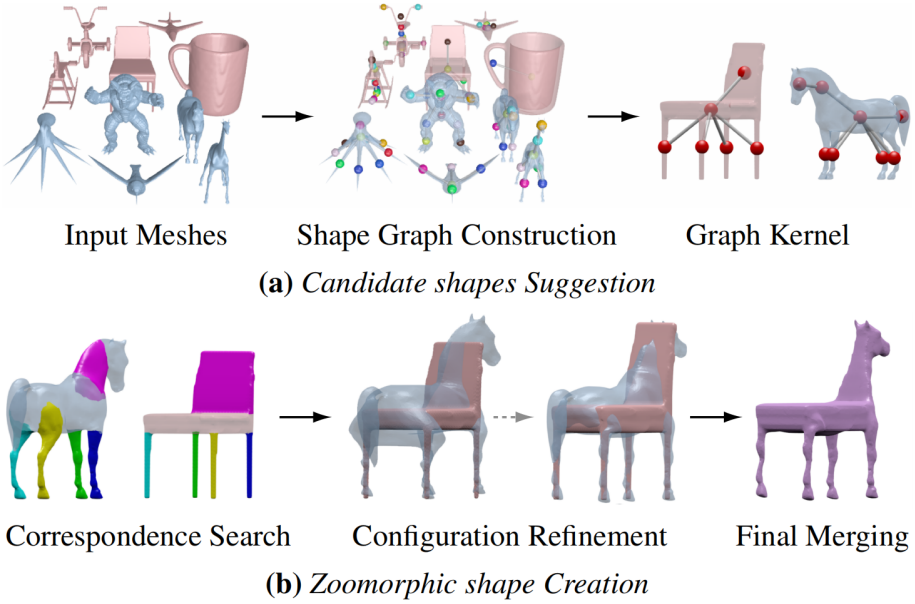


Figure 1.4: Approach overview of [6].

1.1.3 Motivation

Our research motivation is inspired by [13]. Because Liu and Jacobson [13] take into account the polyhedra style, their method can not be directly applied to deform models with smooth, round, and curved surface style, and they need to approximate the round style by polyhedra style. Sphere is the simple curved surface and its surface is smooth, and round. Therefore, we use sphere as the style, and call the smooth, rounder, and curved surface style as spherical style. In 3D world, there are many models which consist of spherical style surfaces. Spherical style surfaces are familiar especially toy models and animation characters, such as Doraemon[1], Kung Fu Panda[2], and Bing Dwen Dwen[3]. Doraemon consists of few spherical style surface components. Both of the head and hands components of Doraemon are spherical style surfaces, also the head and tummy components of Kung Fu Panda. In the aesthetic concept of the public, spherical style surfaces are easily accepted and liked. Therefore, spherical style is meaningful.

[1]Doraemon Channel

[2]Kung Fu Panda (film)

[3]Beijing 2022 Olympic Mascot

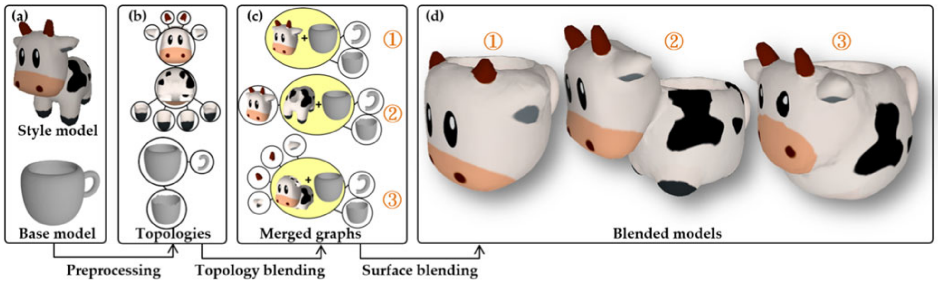


Figure 1.5: Approach overview of [7]. (a) Input models. (b) Topologies of input models. (c-d) Merged graphs and output models.

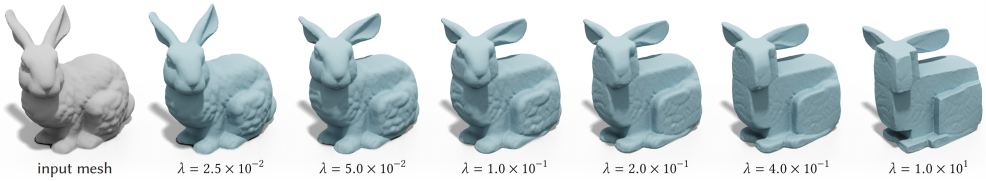


Figure 1.6: Cubic stylization [13]. Cubeness can be controlled by changing the λ parameter.

Because 3D models have complex skeleton structures that consist of many components, when deform a 3D model of multiple components with spherical style, the deformation around connections between each single component are complicated, especially preventing mesh self-intersections. No methods realize spherical style on a 3D model consisting of multiple components. Even spherical style on 3D models, consisting of single component, is also difficult and currently there are no methods, as we know.

1.1.4 Contributions

We intend to use a simple way, such as adjusting parameters, to deform the surfaces with spherical style. In this thesis, we focus on spherical style deformation of single component models and propose a method which deforms the input model with the spherical style, while preserving the local details of the input model. An energy function is defined which combines ARAP and spherical feature. We also provide an efficient optimization solution, and

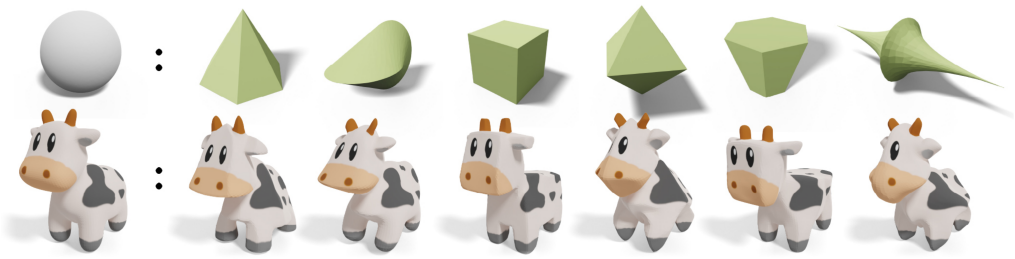


Figure 1.7: Normal-driven spherical shape analogy [14] stylizes an input 3D shape (bottom left) by studying how the surface normal of a style shape (green) relates to the surface normal of a sphere (gray).

apply our deformation method to various input objects. Finally we discuss about the effects of parameters, mesh intersections, and limitations.

The contributions of this thesis are as follows:

(1) We have explored a proper spherical feature and described this feature as ℓ_2-regularization. According to the feature description, energy function is established which combines the ARAP term and the spherical term. An efficient optimization solution is also provided to solve the energy function. To the best of our knowledge, this is the first time that spherical style deformation on single component models has been achieved.

(2) Since the deformation depends on the mesh quality, when the input mesh consists of many obtuse triangles, we also proposed an optional spherical style deformation method based on convex hull proxy model as the complementary deformation method.

1.2 Thesis Outline

This thesis consists of 6 chapters. The outline of the thesis is as follows.

Chapter 1 introduced current research status about stylization of 3D models from local style transfer and global style transfer perspectives. According to the current research status, the research motivation of this thesis was shown, and contributions were summarized.

Chapter 2 discussed about current approaches of global stylization for 3D models. The popular model deformation method as-rigid-as-possible surface

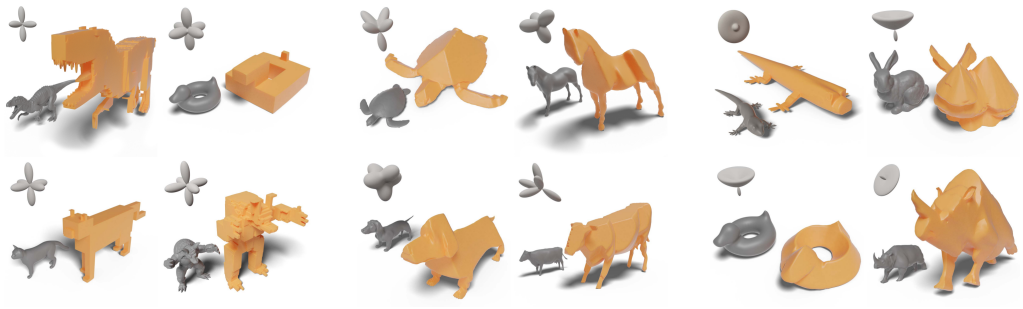


Figure 1.8: A variety of stylization results enabled by Gauss Stylization [11]. Left block: higher values of parameter λ lead to 'extreme' cubification-faces almost parallel to those of a cube. Middle block: Preferred sets of normals are not required to be point-symmetric. Right block: Semi-discrete preferred normals allow modeling locally cylinder-or cone-like shapes.

modeling and related details such as cotangent weight were also introduced.

Chapter 3 described the energy function of spherical style deformation and optimization process for nonlinear spherical feature. However, this method did not succeed. At the same time, we showed discussion about the problems of this mathematical model.

Chapter 4 provided a series of spherical features in a linear form. A proper spherical feature has been explored and the reasons why other features are not suitable are also explained. According to the feature description, energy function is established which combines the ARAP term and the spherical term. An efficient optimization solution is also provided to solve the energy function. The algorithm has performed on a wide variety of single component 3D models, such as convex and smooth models, convex and sharp models, and complex models. For overly growing and mesh self-intersection problems, which are existed in experimental results, we have given explanation and analysis. Finally, limitations of our current algorithm are also discussed.

We also found that the results of our deformation are dependent on the quality of the input mesh. When the input mesh consists of many obtuse triangles, it leads to potential oscillation of the numerical method, poorly conditioned matrices, worsening the speed and accuracy of the linear solver and above spherical style deformation method fails. To solve this problem,

chapter 5 proposed an optional deformation method based on convex hull proxy model as the complementary deformation method. Our proxy method constructs the proxy model of the input model and applies above spherical style deformation method to the proxy model. Finally, deformation result of the input model is obtained by the projection calculation between the proxy model and the input model and interpolation method between the input model and the deformed proxy model. We performed this proxy method to the obtuse triangle mesh and confirmed that the method can achieve better results, such as smoother surface, compared with the method of chapter 4. At the same time, various t functions and partial deformation options give more kinds of deformation possibilities. Finally, we discuss the limitations of the proxy method.

Chapter 6 summarized the work of spherical style deformation on single component model, and proposed the future work.

Chapter 2

Related Works

2.1 Skeleton-based Approach

Skeleton-based approach is a style deformation method based on point-to-point correspondence mapping relationship. Firstly, the method requires complex preprocessing to achieve coarse mapping, such as shape segmentation, skeleton graph construction, correspondence search. Then, mesh parameterization is adopted to obtain point-to-point mapping. Finally, according to the point-to-point mapping relationship, merge surfaces of the two input models and final styled model is achieved. Duncan N and Yu L F [6] firstly perform a series of preprocessing operations, such as segmentation, shape graph construction, and graph kernel. Based on graph kernel technique, they can identify a pair of shapes (a base shape and an animal shape) that are suitable to establish point-to-point mapping. Finally, deform the two shapes jointly by merging method, and obtain the animal styled shape, namely, zoomorphic design. Similar to [6], Huang Y and J Lin [7] add texture information, and performs textural blending to achieve the final styled model.

A series of preprocessing operations can establish the correspondence of components of the two input models and be the basis of point-to-point mapping. Therefore, the skeleton-based approach can deform shapes with complex styles. However, this kind of style deformation method requires much preprocessing to achieve the correspondence relationship of components, compared with the normal-based approach. The skeleton-based approach deforms the

model by complex operations, not in a simple way, such as simply adjusting parameters.

2.2 Normal-based Approach

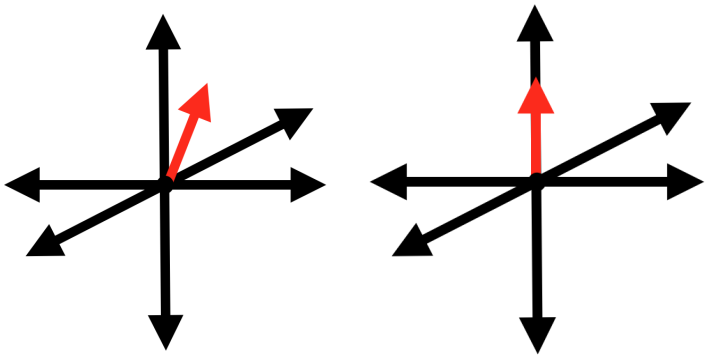
Normal-based approach achieves correspondence mapping relationship in a simple way without complex preprocessing. The method uses the distribution of surface normals to define the desired style, so that the model can be deformed with the desired style.

Liu and Jacobson [13] defined cubic features where the cubic surface normal vectors are parallel to the xyz-axes. They set six target normal directions, namely, the x, -x, y, -y, z, and -z-axis directions. Similarly, using the same idea, Gauss stylization [11] set more target normal directions based on the mixture of Gaussians. Therefore, other than cubic stylization, more stylizations can be generated. Moreover, based on the same idea, Liu and Jacobson [14] developed another method for setting target normal directions, in which the directions were set through a spherical shape analogize to a sphere (the analogy shape), such as spherical parameterization or closest normal method [14].

The essence of [13, 14, 11] is that the original normal vector direction of the point is rotated to the target direction as closely as possible. However, the styles remain mainly as polyhedral styles. They [13, 14, 11] cannot achieve spherical style directly.

Figure 2.1 shows the esstence of mapping correspondence approach through normal vector distribution. In Fig.2.1(a), the six black arrowed lines represent the normal vector distribution of the style model. The red arrowed line represents one normal vector direction of the input model. With this mapping method, the input normal vector (red arrowed line) is compared with the stylized normal vector directions (6 black arrowed lines). The input normal vector will choose the closest stylized normal vector as the target direction, and finally the input model is deformed according to the target direction. After deformation, the direction of the normal vector of the input model is as close as possible to the target direction, shown in Fig.2.1(b). In summary, the mapping correspondence approach through normal vector distribution can not achieve spherical style deformation directly. Because the normal distribution

of sphere is in all directions, shown in Fig.2.2. It is uneffective with above mapping approach to deform models with the normal vector distribution of sphere as style feature.



(a) Normal vector distribution (b) Trend of style deformation

Figure 2.1: Mapping through normal vector distribution

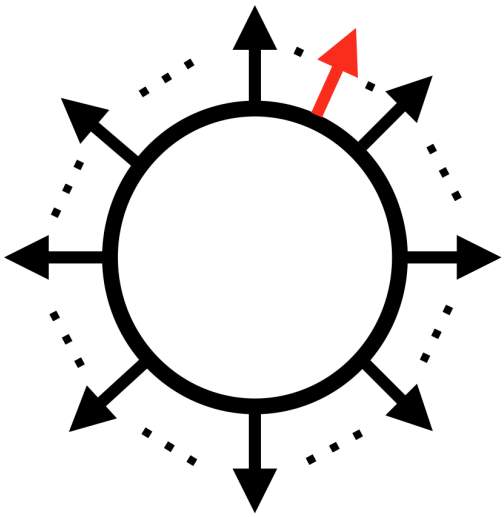


Figure 2.2: Normal vector distribution of sphere

In order to verify above conclusion, we use a polyhedron as the style model to approximate the spherical style, because the distribution of face normals of a polyhedron approximates the normals of a sphere. In the section of experimental results, we show the deformation results performed by the method of [14]. In our research, we define the style of the sphere shape for the first time, and deform the models with the spherical style.

2.3 As-Rigid-As-Possible Surface Modeling

Users prefer deformations can change local shape as rigid as possible [20] so that surface details tend to be preserved. As rigid as possible (ARAP) [20] energy is one popular shape deformation method with constraints, which can preserve local structures of shapes. ARAP decomposes surface into overlapping cells [20]. An ideal deformation seeks to keep the transformation for the surface in each cell as rigid as possible. Overlap of the cells is necessary to avoid surface stretching or shearing at the boundary of the cells [20]. Each cell covers the triangles incident upon a point (i.e. the one ring neighborhood). In this thesis, each cell uses half-edge data structure, including spokes and rims [12, 3, 13], shown as Fig.2.3.

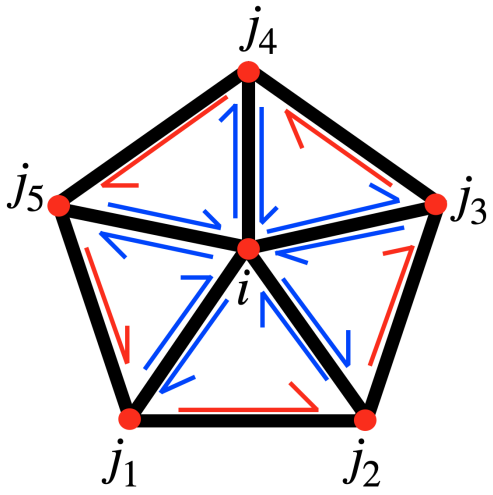


Figure 2.3: Cell structure

In Fig.2.3, the center point is the ith point, and its one-ring neighborhood includes j_1th, j_2th, j_3th, j_4, and j_5th points. The blue lines with arrow represent spokes, and the red lines with arrow represent rims. Sorkine and Alexa [20] define only spokes as the cell C_i. Every point of the input mesh model forms the coresponding cell.

Given the cell C_i corresponding to the ith point that deforms to C_i', when the transformation is rigid, a rotation $\mathbf{R}_i$ satisfies Eq.2.1 [20]. $\mathbf{p}_i$ and $\mathbf{p}_i'$ are positions of the ith point before and after deformation. $N(i)$ is the one ring

neighborhood set of the ith point.

$$\mathbf{p}'_i - \mathbf{p}'_j = \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j), \forall j \in N(i) \tag{2.1}$$

When the transformation of the cell C_i is not completely rigid, minimize the energy function $E(C_i, C'_i)$ (shown as Eq.2.2) to obtain the maximized rigid transformation $\mathbf{R}_i$ [20]. w_{ij} is the cotangent weight introduced in section 2.4.

$$E(C_i, C'_i) = \sum_{j \in N(i)} w_{ij} ||(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 \tag{2.2}$$

Summing up energy functions from all cells, a global energy function is as Eq.2.3 [20].

$$\begin{aligned} E(S') &= \sum_{i=1}^{n} E(C_i, C'_i) \\ &= \sum_{i=1}^{n} \sum_{j \in N(i)} w_{ij} ||(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 \end{aligned} \tag{2.3}$$

In chapter 3 of this thesis, we focus on the optimization solution, and the definition of cell C_i is same as Sorkine and Alexa [20]. In chapter 4 of this thesis, all spokes and rims form the cell C_i (shown as Fig.2.3). The comparision of spokes and rims as a cell and only spokes as a cell is shown in Figure 8 of [8]. Spokes-only ARAP energy produces artifacts due to indefinite terms in the energy function [8].

2.4 Cotangent Weight

Cotangent weight can maintain the geometry property better than uniform weight [20]. The calculation of cotangent weights is referred [18, 16]. Point $\mathbf{p}$ and $\mathbf{q}_1$ are shown in Fig.2.4. w_{pq1} is the cotangent weight of the edge $\mathbf{e}_{pq1}$. w_{pq1} is as Eq.2.4.

$$w_{pq1} = \frac{1}{2}(\cot \alpha_1 + \cot \beta_1) \tag{2.4}$$

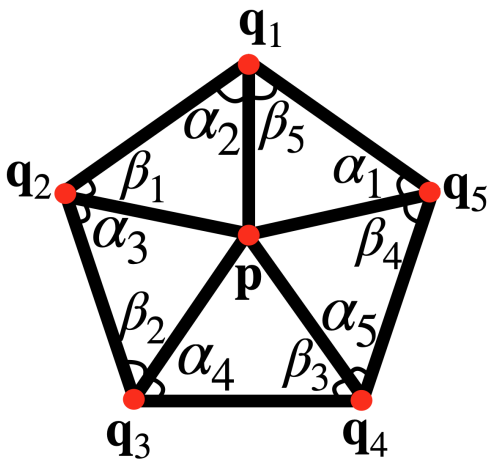


Figure 2.4: Neighborhood around a point [18]

$$\begin{aligned}\cot\alpha_1 &= \frac{\cos\alpha_1}{\sin\alpha_1} \\ &= \frac{\mathbf{q_5p}\cdot\mathbf{q_5q_1}}{\mathbf{q_5p}\times\mathbf{q_5q_1}} \\ &= \frac{-\mathbf{pq_5}\cdot\mathbf{q_5q_1}}{\mathbf{pq_5}\times\mathbf{q_5q_1}} \\ &= \frac{-\mathbf{pq_5}\cdot\mathbf{q_5q_1}}{2S_\triangle}\end{aligned} \tag{2.5}$$

$S_\triangle$ is the area value of the triangle pq_5q_1. $\mathbf{q_5p}$ is the vector from the point $\mathbf{q}_5$ to the point $\mathbf{p}$.

Chapter 3

Nonlinear Spherical Feature and Optimization Process

3.1 Overview

We define an energy function that combines ARAP and Spherical Feature, similar with cubic stylization [13]. Our approach retains the ARAP term and replaces the cubeness, which aligns the rotated point normals with the xyz-axes, with the spherical part which is represented as the difference between the rotated point normal vectors and the "outward" vectors from the spherical center to positions.

We describe the symbols used in this paper. Let M be the input original mesh, $\{i|i \in N\} = \mathcal{V}$ be the set of points indices and $\mathbf{V} = \{\mathbf{p}_i\}$ be the set of point positions. Edges and triangle faces are denoted as $(i, j) \in \mathcal{T}$ and $(i, j, k) \in \mathcal{F}$, respectively. We denote the deformed mesh as M' and point positions as $\mathbf{V}' = \{\mathbf{p}'_i\}$. Note that our deformation does not change the mesh topology, $\mathcal{T}$ and $\mathcal{F}$ are invariant. Point position $\mathbf{p}'_i$ is a 3×1 vector. In section 3.4.1, vector of $\mathbf{p}'_i$ in matrix form is transposed by default.

3.2 Nonlinear Spherical Feature

In this section, we explain how to define the evaluation function for spherical feature. We would like to evaluate spherical feature using normal vectors in this section. The normal vector direction of a point on the spherical surface is the same as the vector direction from the spherical center to the point position on the spherical surface. According to this spherical feature, the energy is minimized as Eq.3.1:

$$\begin{aligned}\underset{\mathbf{V}',\{\mathbf{R}_i\}}{\text{minimize}} \sum_{i\in\mathcal{V}} \sum_{j\in N(i)} w_{ij}||(\mathbf{p}'_i-\mathbf{p}'_j)-\mathbf{R}_i(\mathbf{p}_i-\mathbf{p}_j)||^2+ \\ \lambda \sum_{j\in N(i)'} ||\mathbf{R}_i\mathbf{n}_j-\frac{(\mathbf{p}'_j-\mathbf{o})}{||\mathbf{p}'_j-\mathbf{o}||}||^2\end{aligned} \tag{3.1}$$

In Eq.3.1: The first term is the ARAP energy [20]. The second term is the spherical energy. Minimizing the ARAP energy encourages to preserve the original shape. λ is a parameter controlling the spherical style deformation, such as the roundness of a shape. $\mathbf{p}'_i$ and $\mathbf{p}_i$ are the deformed and original ith point positions respectively; $\mathbf{o}$ is the center position of the sphere; $\mathbf{R}_i$ is the rotation matrix; $N(i)$ is the one ring neighbors of the ith point, not including the ith point; $N(i)'$ is the one ring neighbors of the ith point, including the ith point; $\mathbf{n}_j$ is a 3×1 unit normal vector of the original model; w_{ij} is the cotangent weight [18, 16]. Minimizing the spherical energy encourages the one ring neighbors of the ith point to satisfy the spherical feature, such as rounder surface.

However, the part $||\mathbf{R}_i\mathbf{n}_j-\frac{(\mathbf{p}'_j-\mathbf{o})}{||\mathbf{p}'_j-\mathbf{o}||}||^2$ in Eq.3.1 is non-linear about the variable $\mathbf{p}'_j$, and makes the optimization difficult. Therefore, we tried to introduce an auxiliary variable s_j for local optimization, and transform to:

$$\begin{aligned}\underset{\mathbf{V}',\{\mathbf{R}_i\}}{\text{minimize}} \sum_{i\in\mathcal{V}} \sum_{j\in N(i)} w_{ij}||(\mathbf{p}'_i-\mathbf{p}'_j)-\mathbf{R}_i(\mathbf{p}_i-\mathbf{p}_j)||^2+ \\ \lambda \sum_{j\in N(i)'} ||\mathbf{R}_i\mathbf{n}_j-s_j(\mathbf{p}'_j-\mathbf{o})||^2\end{aligned} \tag{3.2}$$

In Eq.3.2, s_j is the reciprocal of the length of $(\mathbf{p}'_j-\mathbf{o})$, and the part $||\mathbf{R}_i\mathbf{n}_j-s_j(\mathbf{p}'_j-\mathbf{o})||^2$ become linear. Next, I will introduce how to optimize the energy function of Eq.3.2.

Local-global optimization strategy are tried to solve the energy function of Eq.3.2. Next we will describe the optimization process: local step optimization and global step optimization.

3.3 Local Step Optimization

In local step optimization, the idea is that given positions of points, solve the optimal rigid transformations: rotation matrix $\mathbf{R}_i$ and auxiliary variable s_j. Given the cell C_i, which covers the triangles incident upon the ith point, and its deformed cell C_i' [20], the energy related to C_i and C_i' is as Eq.3.3:

$$\begin{aligned} E(C_i, C_i') = & \sum_{j \in N(i)} w_{ij} ||(\mathbf{p}_i' - \mathbf{p}_j') - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 \\ & + \lambda \sum_{j \in N(i)'} ||\mathbf{R}_i \mathbf{n}_j - s_j(\mathbf{p}_j' - \mathbf{o})||^2 \end{aligned} \tag{3.3}$$

3.3.1 Optimization of Rotation Matrix $\mathbf{R}_i$

Given point positions $\{\mathbf{p}_i\}$, $\{\mathbf{p}_i'\}$, solve the rotation matrix $\mathbf{R}_i$, when minimizing the energy $E(C_i, C_i')$. Minimize the energy of $E(C_i, C_i')$, and expand Eq.3.3 as Eq.3.4. $\mathbf{e}_{ij} = \mathbf{p}_i - \mathbf{p}_j$.

$$\begin{aligned} & \underset{\mathbf{R}_i}{\text{minimize}} E(C_i, C_i') \\ = & \underset{\mathbf{R}_i}{\text{minimize}} \sum_{j \in N(i)} w_{ij} (\mathbf{e}_{ij}' - \mathbf{R}_i \mathbf{e}_{ij})^{\mathrm{T}} (\mathbf{e}_{ij}' - \mathbf{R}_i \mathbf{e}_{ij}) \\ & + \lambda \sum_{j \in N(i)'} (\mathbf{R}_i \mathbf{n}_j - s_j(\mathbf{p}_j' - \mathbf{o}))^{\mathrm{T}} (\mathbf{R}_i \mathbf{n}_j - s_j(\mathbf{p}_j' - \mathbf{o})) \\ = & \underset{\mathbf{R}_i}{\text{minimize}} \sum_{j \in N(i)} w_{ij} ({\mathbf{e}_{ij}'}^{\mathrm{T}} \mathbf{e}_{ij}' + {\mathbf{e}_{ij}}^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_i \mathbf{e}_{ij} - 2{\mathbf{e}_{ij}'}^{\mathrm{T}} \mathbf{R}_i \mathbf{e}_{ij}) \\ & + \lambda \sum_{j \in N(i)'} (\mathbf{n}_j^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_i \mathbf{n}_j + s_j^2 (\mathbf{p}_j' - \mathbf{o})^{\mathrm{T}} (\mathbf{p}_j' - \mathbf{o}) - 2 s_j (\mathbf{p}_j' - \mathbf{o})^{\mathrm{T}} \mathbf{R}_i \mathbf{n}_j) \end{aligned} \tag{3.4}$$

Delete the constant terms, and the formula is as Eq.3.5 :

$$\underset{\mathbf{R}_i}{\text{minimize}} \sum_{j \in N(i)} -2 w_{ij} ({\mathbf{e}_{ij}'}^{\mathrm{T}} \mathbf{R}_i \mathbf{e}_{ij}) + \lambda \sum_{j \in N(i)'} (-2) s_j (\mathbf{p}_j' - \mathbf{o})^{\mathrm{T}} \mathbf{R}_i \mathbf{n}_j \tag{3.5}$$

Equation 3.5 is equivalent to Eq.3.6:

$$\begin{aligned} & \arg\max_{\mathbf{R}_i} \sum_{j \in N(i)} w_{ij} {\mathbf{e}_{ij}'}^{\mathrm{T}} \mathbf{R}_i \mathbf{e}_{ij} + \lambda \sum_{j \in N(i)'} s_j (\mathbf{p}_j' - \mathbf{o})^{\mathrm{T}} \mathbf{R}_i \mathbf{n}_j \\ = & \arg\max_{\mathbf{R}_i} \mathrm{tr}(\sum_{\mathrm{j} \in \mathrm{N(i)}} \mathrm{w_{ij}} \mathbf{R}_\mathrm{i} \mathbf{e}_\mathrm{ij} {\mathbf{e}_\mathrm{ij}'}^{\mathrm{T}} + \lambda \sum_{\mathrm{j} \in \mathrm{N(i)'}} \mathrm{s_j} \mathbf{R}_\mathrm{i} \mathbf{n}_\mathrm{j} (\mathbf{p}_\mathrm{j}' - \mathbf{o})^{\mathrm{T}}) \\ = & \arg\max_{\mathbf{R}_i} \mathrm{tr}(\mathbf{R}_\mathrm{i} (\sum_{\mathrm{j} \in \mathrm{N(i)}} \mathrm{w_{ij}} \mathbf{e}_\mathrm{ij} {\mathbf{e}_\mathrm{ij}'}^{\mathrm{T}} + \lambda \sum_{\mathrm{j} \in \mathrm{N(i)'}} \mathrm{s_j} \mathbf{n}_\mathrm{j} (\mathbf{p}_\mathrm{j}' - \mathbf{o})^{\mathrm{T}})) \\ = & \arg\max_{\mathbf{R}_i} \mathrm{tr}(\mathbf{R}_\mathrm{i} \mathbf{S}_\mathrm{i}) \end{aligned} \tag{3.6}$$

In Eq.3.6,

$$\mathbf{S}_i = \sum_{j \in N(i)} w_{ij} \mathbf{e}_{ij} {\mathbf{e}'_{ij}}^{\mathrm{T}} + \lambda \sum_{j \in N(i)'} s_j \mathbf{n}_j (\mathbf{p}'_j - \mathbf{o})^{\mathrm{T}} \tag{3.7}$$

Decompose $\mathbf{S}_i$ with singular value decomposition as Eq.3.8:

$$\mathbf{S}_i = \mathbf{U}_i \sum\nolimits_i \mathbf{V}_i^{\mathrm{T}} \tag{3.8}$$

Rotation matrix $\mathbf{R}_i$ is as Eq.3.9:

$$\mathbf{R}_i = \mathbf{V}_i \mathbf{U}_i^{\mathrm{T}} \tag{3.9}$$

If $det(\mathbf{R}_i) < 0$, the last column of $\mathbf{U}_i$ needs to be negated.

3.3.2 Optimization of Auxiliary Variable s_j

After solving the rotation matrix $\mathbf{R}_i$, fix $\mathbf{R}_i$, and solve auxiliary variable s_j. In Eq.3.3, the energy function is a quadratic function with respect to the auxiliary variable s_j. The term of s_j is $\lambda \sum_{j \in N(i)'} ||\mathbf{R}_i \mathbf{n}_j - s_j(\mathbf{p}'_j - \mathbf{o})||^2$. The solver process of s_j is as Eq.3.10:

$$\begin{aligned}
&\underset{s_j}{\text{minimize}} \quad \lambda \sum_{j \in N(i)'} ||\mathbf{R}_i \mathbf{n}_j - s_j(\mathbf{p}'_j - \mathbf{o})||^2 \\
=\ &\underset{s_j}{\text{minimize}} \quad \lambda \sum_{j \in N(i)'} [\mathbf{R}_i \mathbf{n}_j - s_j(\mathbf{p}'_j - \mathbf{o})]^{\mathrm{T}} [\mathbf{R}_i \mathbf{n}_j - s_j(\mathbf{p}'_j - \mathbf{o})] \\
=\ &\underset{s_j}{\text{minimize}} \quad \lambda \sum_{j \in N(i)'} [\mathbf{n}_j^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_i \mathbf{n}_j + s_j^2 (\mathbf{p}'_j - \mathbf{o})^{\mathrm{T}} (\mathbf{p}'_j - \mathbf{o}) - 2 s_j (\mathbf{p}'_j - \mathbf{o})^{\mathrm{T}} \mathbf{R}_i \mathbf{n}_j]
\end{aligned} \tag{3.10}$$

Delete constant terms as Eq.3.11:

$$\underset{s_j}{\text{minimize}} \quad \lambda \sum_{j \in N(i)'} [s_j^2 (\mathbf{p}'_j - \mathbf{o})^{\mathrm{T}} (\mathbf{p}'_j - \mathbf{o}) - 2 s_j (\mathbf{p}'_j - \mathbf{o})^{\mathrm{T}} \mathbf{R}_i \mathbf{n}_j] \tag{3.11}$$

In Eq.3.11, the term about auxiliary variable s_j is quadratic. The minimum value of s_j is as Eq.3.12:

$$s_j = \frac{\sum\limits_{j \in N(i)'} (\mathbf{p}'_j - \mathbf{o})^{\mathrm{T}} \mathbf{R}_i \mathbf{n}_j}{\sum\limits_{j \in N(i)'} (\mathbf{p}'_j - \mathbf{o})^{\mathrm{T}} (\mathbf{p}'_j - \mathbf{o})} \tag{3.12}$$

3.4 Global Step Optimization

The total energy of the input mesh is to add each cell energy $E(C_i, C_i')$. Therefore we obtain the total energy function as Eq.3.13 and the number of points is n.

$$E(M') = \sum_{i=1}^{n} E(C_i, C_i') \tag{3.13}$$

Given $\{\mathbf{R}_i\}$, solve point positions $\mathbf{p}_i$. In Eq.3.13, the total energy function is a quadratic function with respect to $\{\mathbf{p}_i\}$. The solver process is as follows. Firstly, represent the system of equations in matrix form. Secondly, solve for positions $\{\mathbf{p}_i\}$ by matrix factorization method.

3.4.1 Representation of Linear System

The derivative of the energy function Eq.3.13 with respect to $\mathbf{p}_i$ is as Eq.3.14.

$$\begin{aligned}
&\frac{\partial E(M')}{\partial \mathbf{p}_i'} \\
&= \frac{\partial}{\partial \mathbf{p}_i'}\{\sum_{j\in N(i)} w_{ij}||(\mathbf{p}_i' - \mathbf{p}_j') - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 + \lambda \sum_{j\in N(i)'} ||\mathbf{R}_i\mathbf{n}_j - s_j(\mathbf{p}_j' - \mathbf{o})||^2 \\
&\quad + \sum_{j\in N(i)} w_{ji}||(\mathbf{p}_j' - \mathbf{p}_i') - \mathbf{R}_j(\mathbf{p}_j - \mathbf{p}_i)||^2 + \lambda \sum_{j\in N(i)'} ||\mathbf{R}_j\mathbf{n}_i - s_i(\mathbf{p}_i' - \mathbf{o})||^2\} \\
&= \sum_{j\in N(i)} 2w_{ij}[(\mathbf{p}_i' - \mathbf{p}_j') - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)] + 2\lambda[\mathbf{R}_i\mathbf{n}_i - s_i(\mathbf{p}_i' - \mathbf{o})](-s_i) \\
&\quad + \sum_{j\in N(i)} -2w_{ji}[(\mathbf{p}_j' - \mathbf{p}_i') - \mathbf{R}_j(\mathbf{p}_j - \mathbf{p}_i)] + \sum_{j\in N(i)} -2\lambda s_i[\mathbf{R}_j\mathbf{n}_i - s_i(\mathbf{p}_i' - \mathbf{o})] \\
&= \sum_{j\in N(i)'} \{2w_{ij}[(\mathbf{p}_i' - \mathbf{p}_j') - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)] + 2w_{ij}[(\mathbf{p}_i' - \mathbf{p}_j') - \mathbf{R}_j(\mathbf{p}_i - \mathbf{p}_j)]\} \\
&\quad + 2\lambda s_i \sum_{j\in N(i)'} [s_i(\mathbf{p}_i' - \mathbf{o}) - \mathbf{R}_j\mathbf{n}_i] \\
&= \sum_{j\in N(i)'} 4w_{ij}[(\mathbf{p}_i' - \mathbf{p}_j') - \frac{(\mathbf{R}_i + \mathbf{R}_j)}{2}(\mathbf{p}_i - \mathbf{p}_j)] \\
&\quad + 2\lambda s_i \sum_{j\in N(i)'} [s_i(\mathbf{p}_i' - \mathbf{o}) - \mathbf{R}_j\mathbf{n}_i]
\end{aligned} \tag{3.14}$$

Since the weights are symmetric, $w_{ij} = w_{ji}$. In order to achieve the optimal value of $\mathbf{p}_i$, set Eq.3.14 to zero, and obtain Eq.3.15:

$$\begin{aligned} &\sum_{j \in N(i)'} 4w_{ij}(\mathbf{p}'_i - \mathbf{p}'_j) + 2\lambda s_i \sum_{j \in N(i)'} s_i(\mathbf{p}'_i - \mathbf{o}) \\ &= \sum_{j \in N(i)'} 4w_{ij} \times \frac{(\mathbf{R}_i + \mathbf{R}_j)}{2}(\mathbf{p}_i - \mathbf{p}_j) + 2\lambda s_i \sum_{j \in N(i)'} \mathbf{R}_j \mathbf{n}_i \end{aligned} \tag{3.15}$$

Reduce both sides of the equation by a factor of 4 as Eq.3.16

$$\begin{aligned} &\sum_{j \in N(i)'} [w_{ij}(\mathbf{p}'_i - \mathbf{p}'_j) + \frac{1}{2}\lambda s_i^2 \mathbf{p}'_i - \frac{1}{2}\lambda s_i^2 \mathbf{o}] \\ &= \sum_{j \in N(i)'} [\frac{w_{ij}}{2}(\mathbf{R}_i + \mathbf{R}_j)(\mathbf{p}_i - \mathbf{p}_j) + \frac{1}{2}\lambda s_i \mathbf{R}_j \mathbf{n}_i] \end{aligned} \tag{3.16}$$

Organize Eq.3.16 to get Eq.3.17:

$$\begin{aligned} &\sum_{j \in N(i)'} [(w_{ij} + \frac{1}{2}\lambda s_i^2)\mathbf{p}'_i] + \sum_{j \in N(i)'} (-w_{ij}\mathbf{p}'_j) + \sum_{j \in N(i)'} (-\frac{1}{2}\lambda s_i^2 \mathbf{o}) \\ &= \sum_{j \in N(i)'} [\frac{w_{ij}}{2}(\mathbf{R}_i + \mathbf{R}_j)(\mathbf{p}_i - \mathbf{p}_j) + \frac{1}{2}\lambda s_i \mathbf{R}_j \mathbf{n}_i] \end{aligned} \tag{3.17}$$

Note that sphere center is as Eq.3.18:

$$\mathbf{o} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}'_i \tag{3.18}$$

Firstly, we think $\{\mathbf{p}'_1, \mathbf{p}'_2, ..., \mathbf{p}'_i, ..., \mathbf{p}'_n, \mathbf{o}\}$ are variables. Equation 3.19 is the matrix form of Eq.3.17.

$$\mathbf{L}_{non} \times \mathbf{H}_0 = \mathbf{B}_{non} \tag{3.19}$$

$\mathbf{H}_0$ represents the variables $\{\mathbf{p}'_1, \mathbf{p}'_2, ..., \mathbf{p}'_i, ..., \mathbf{p}'_n, \mathbf{o}\}$ shown in Eq.3.20. The size of $\mathbf{H}_0$ is $(n+1) \times 3$. Point position $\mathbf{p}'_i$ is a 3×1 vector. In Eq.3.20, vector of $\mathbf{p}'_i$ is transposed by default.

$$\mathbf{H_0} = \begin{bmatrix} \mathbf{p}'_1 & \mathbf{p}'_2 & \cdots & \mathbf{p}'_i & \cdots & \mathbf{p}'_n & \mathbf{o} \end{bmatrix}^{\mathrm{T}} \tag{3.20}$$

In Eq.3.17, we represent the right side of the equal sign as the matrix form $\mathbf{B}_{non}$. The size of $\mathbf{B}_{non}$ is $n \times 3$. Equation 3.21 shows $\mathbf{B}_{non}$.

$$\mathbf{B}_{non} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_i & \cdots & \mathbf{b}_{n-1} & \mathbf{b}_n \end{bmatrix}^{\mathrm{T}} \tag{3.21}$$

$\mathbf{b}_i$ is a 1×3 vector shown as Eq.3.22.

$$\mathbf{b}_i = \sum_{j \in N(i)'} [\frac{w_{ij}}{2}(\mathbf{R}_i + \mathbf{R}_j)(\mathbf{p}_i - \mathbf{p}_j) + \frac{1}{2}\lambda s_i \mathbf{R}_j \mathbf{n}_i]^{\mathrm{T}} \tag{3.22}$$

According to Eq.3.17, for every point of the whole mesh, there is a linear system of equations. $\mathbf{L}_{non}$ is the coefficient matrix of the linear system of equations shown as Eq.3.23.

$$\begin{bmatrix} W_1 + g_1 & -w_{12} & \cdots & -w_{1i} & \cdots & -w_{1,n-1} & -w_{1,n} & -g_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{i,1} & -w_{i,2} & \cdots & W_i + g_i & \cdots & -w_{i,n-1} & -w_{i,n} & -g_i \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{n-1,1} & -w_{n-1,2} & \cdots & -w_{n-1,i} & \cdots & W_{n-1} + g_{n-1} & -w_{n-1,n} & -g_{n-1} \\ -w_{n,1} & -w_{n,2} & \cdots & -w_{n,i} & \cdots & -w_{n,n-1} & W_n + g_n & -g_n \end{bmatrix} \tag{3.23}$$

W_i is used to represent as Eq.3.24.

$$W_i = \sum_{j \in N(i)'} w_{ij} \tag{3.24}$$

g_i is used to represent as Eq.3.25. nu_i is the number of points in the one-ring neighborhood of ith point, including the ith point.

$$g_i = \frac{1}{2}\lambda s_i^2 nu_i \tag{3.25}$$

The size of matrix $\mathbf{L}_{non}$ is $n \times (n+1)$. Considering the first n rows and first n columns of the matrix $\mathbf{L}_{non}$ as a sub-matrix, the sub-matrix is sparse and symmetric.

According to Eq.3.18, we can replace one variable, and only have n variables. We replace $\mathbf{p}'_n$ with $\{\mathbf{p}'_1, \mathbf{p}'_2, ..., \mathbf{p}'_i, ..., \mathbf{p}'_{n-1}, \mathbf{o}\}$ shown as Eq.3.26.

$$\mathbf{p}'_n = n\mathbf{o} - \sum_{k=1}^{n-1} \mathbf{p}'_k \tag{3.26}$$

Variables also change into $\mathbf{H}_1$ shown as Eq.3.27

$$\mathbf{H}_1 = \begin{bmatrix} \mathbf{p}'_1 & \mathbf{p}'_2 & \cdots & \mathbf{p}'_i & \cdots & \mathbf{p}'_{n-1} & \mathbf{o} \end{bmatrix}^{\mathrm{T}} \tag{3.27}$$

$-w_{in}\mathbf{p}'_n$ is represented by $\mathbf{H}_1$ shown as Eq.3.28:

$$\begin{aligned} -w_{in}\mathbf{p}'_n =& -w_{in}(n\mathbf{o} - \sum_{k=1}^{n-1} \mathbf{p}'_k) \\ =& -w_{in} \times n\mathbf{o} + w_{in}\mathbf{p}'_1 + w_{in}\mathbf{p}'_2 + \cdots + w_{in}\mathbf{p}'_{n-1} \end{aligned} \tag{3.28}$$

$\mathbf{L}_{non}(n,n)$ is represented by $\mathbf{H}_1$ shown as Eq.3.29:

$$\begin{aligned}&(W_n+g_n)\mathbf{p}'_n\\ &=(W_n+g_n)(n\mathbf{o}-\sum_{k=1}^{n-1}\mathbf{p}'_k)\\ &=(W_n+g_n)\times n\mathbf{o}+(W_n+g_n)(-1)(\mathbf{p}'_1+\mathbf{p}'_2+\cdots+\mathbf{p}'_{n-1})\end{aligned} \tag{3.29}$$

Then the matrix $\mathbf{L}_{non}$ can be changed to matrix $\mathbf{L}_{non1}$ shown as Eq.3.30. The scale of $\mathbf{L}_{non1}$ is $n \times n$.

$$\begin{bmatrix} W_1+g_1+w_{1n} & \cdots & -w_{1i}+w_{1n} & \cdots & -w_{1,n-1}+w_{1n} & -g_1-nw_{1n}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ -w_{i,1}+w_{i,n} & \cdots & W_i+g_i+w_{i,n} & \cdots & -w_{i,n-1}+w_{i,n} & -g_i-nw_{i,n}\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\ -w_{n-1,1}+w_{n-1,n} & \cdots & -w_{n-1,i}+w_{n-1,n} & \cdots & W_{n-1}+g_{n-1}+w_{n-1,n} & -g_{n-1}-nw_{n-1,n}\\ -w_{n,1}-W_n-g_n & \cdots & -w_{n,i}-W_n-g_n & \cdots & -w_{n,n-1}-W_n-g_n & -g_n+n(W_n+g_n)\end{bmatrix} \tag{3.30}$$

If $n \notin N(i)'$, $w(i,n)=0$. Therefore $\mathbf{L}_{non1}$ is a sparse, but not symmetric matrix. $\mathbf{L}_{non1}$ still satisfies Eq.3.31.

$$\mathbf{L}_{non1}\times\mathbf{H}_1=\mathbf{B}_{non} \tag{3.31}$$

3.4.2 Solver of Linear System

$\mathbf{L}_{non1}$ is a sparse matrix. We fix sphere center $\mathbf{o}$. At first, we transform the center $\mathbf{o}$ of the input mesh model to $[0 \quad 0 \quad 0]$. Then we fix $\mathbf{o}$, that is, $\mathbf{o}=[0 \quad 0 \quad 0]$, and solve variables $\{\mathbf{p}'_1,\mathbf{p}'_2,...,\mathbf{p}'_i,...,\mathbf{p}'_{n-1}\}$. Equation 3.31 transforms into Eq.3.32.

$$\mathbf{L}_{non1}\times(\mathbf{H}_2+\mathbf{H}_{fix})=\mathbf{B}_{non} \tag{3.32}$$

$\mathbf{H}_2$ and $\mathbf{H}_{fix}$ satisfy the relationship as Eq.3.33. $\mathbf{o}_1=[0 \quad 0 \quad 0]$.

$$\begin{aligned}\mathbf{H}_1=&\mathbf{H}_2+\mathbf{H}_{fix}\\ =&\begin{bmatrix}\mathbf{p}'_1\\ \mathbf{p}'_2\\ \vdots\\ \mathbf{p}'_i\\ \vdots\\ \mathbf{p}'_{n-1}\\ \mathbf{o}_1\end{bmatrix}+\begin{bmatrix}\mathbf{o}_1\\ \mathbf{o}_1\\ \vdots\\ \mathbf{o}_1\\ \vdots\\ \mathbf{o}_1\\ \mathbf{o}\end{bmatrix}\end{aligned} \tag{3.33}$$

Equation 3.32 transforms to Eq.3.34.

$$\mathbf{L}_{non1} \times \mathbf{H}_2 = \mathbf{B}_{non} - \mathbf{L}_{non1} \times \mathbf{H}_{fix} \tag{3.34}$$

$\mathbf{o}_1$ is fixed values in $\mathbf{H}_2$. Therefore, delete the coefficients corresponding to $\mathbf{o}_1$ in the matrix $\mathbf{L}_{non1}$, that is, delete the nth column of $\mathbf{L}_{non1}$. Because now the number of variables is $(n-1)$, n equations are not necessary. The nth row of $\mathbf{L}_{non1}$ is also deleted. After deletion of $\mathbf{L}_{non1}$, the matrix is represented as $\mathbf{L}_{non2}$ shown as Eq.3.35.

$$\begin{bmatrix} W_1 + g_1 + w_{1n} & \cdots & -w_{1i} + w_{1n} & \cdots & -w_{1,n-1} + w_{1n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{i,1} + w_{i,n} & \cdots & W_i + g_i + w_{i,n} & \cdots & -w_{i,n-1} + w_{i,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{n-1,1} + w_{n-1,n} & \cdots & -w_{n-1,i} + w_{n-1,n} & \cdots & W_{n-1} + g_{n-1} + w_{n-1,n} \end{bmatrix} \tag{3.35}$$

Similarly, after deletion of the nth row of $\mathbf{H}_2$, $\mathbf{H}_3$ is used to represent the $(n-1)$ variables shown as Eq.3.36.

$$\mathbf{H}_3 = \begin{bmatrix} \mathbf{p}'_1 & \mathbf{p}'_2 & \cdots & \mathbf{p}'_i & \cdots & \mathbf{p}'_{n-1} \end{bmatrix}^{\mathrm{T}} \tag{3.36}$$

At the same time, the nth row of the right side of the equal sign in Eq.3.34 is also deleted, and use $\mathbf{B}_{non1}$ to represent the right side. The linear system about variables $\{\mathbf{p}'_1, \mathbf{p}'_2, ..., \mathbf{p}'_i, ..., \mathbf{p}'_{n-1}\}$ is as Eq.3.37.

$$\mathbf{L}_{non2} \times \mathbf{H}_3 = \mathbf{B}_{non1} \tag{3.37}$$

$\mathbf{L}_{non2}$ is also a sparse, but not symmetrical matrix. The size of $\mathbf{L}_{non2}$ is $(n-1) \times (n-1)$.

Next we will introduce the solver process of $\mathbf{H}_3$. Firstly, LU Factorization of $\mathbf{L}_{non2}$ is adopted as Eq.3.38. $\mathbf{L}_{lu}$ is lower triangular matrix, and $\mathbf{U}_{lu}$ is upper triangular matrix.

$$\mathbf{L}_{non2} = \mathbf{L}_{lu}\mathbf{U}_{lu} \tag{3.38}$$

Secondly, $\mathbf{H}_3$ satisfies the formula as Eq.3.39.

$$\begin{cases} \mathbf{X}_1 = \mathbf{U}_{lu} \times \mathbf{H}_3 \\ \mathbf{L}_{lu} \times \mathbf{X}_1 = \mathbf{B}_{n1} \end{cases} \tag{3.39}$$

From Eq.3.39, we can solve $\mathbf{X}_1$ according to Eq.3.40.

$$\mathbf{X}_1 = {\mathbf{L}_{lu}}^{-1} \times \mathbf{B}_{non1} \tag{3.40}$$

Finally, $\mathbf{H}_3$ is solved by Eq.3.41.

$$\mathbf{H}_3 = {\mathbf{U}_{lu}}^{-1} \times \mathbf{X}_1 \tag{3.41}$$

Due to $\mathbf{o} = [0 \quad 0 \quad 0]$, $\mathbf{p}'_n$ is as Eq.3.42.

$$\mathbf{p}'_n = -\sum_{k=1}^{n-1} \mathbf{p}'_k \tag{3.42}$$

Therefore, variables $\{\mathbf{p}'_1, \mathbf{p}'_2, ..., \mathbf{p}'_i, ..., \mathbf{p}'_{n-1}, \mathbf{p}'_n\}$ are obtained by Eq.3.41 and 3.42.

3.5 Optimization Stopping Criteria

Point positions of the input mesh model denote as $\mathbf{P}$ shown in Eq.3.43.

$$\mathbf{P} = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \cdots \quad \mathbf{p}_i \quad \cdots \quad \mathbf{p}_{n-1} \quad \mathbf{p}_n]^{\mathrm{T}} \tag{3.43}$$

$\mathbf{U}_i$ and $\mathbf{U}_{i-1}$ are respectively denoted as point positions of the ith and $(i-1)$th iterations of global optimization shown in Eq.3.44 and 3.45.

$$\mathbf{U}_i = \left[\mathbf{p}_1^{i'} \quad \mathbf{p}_2^{i'} \quad \cdots \quad \mathbf{p}_i^{i'} \quad \cdots \quad \mathbf{p}_{n-1}^{i'} \quad \mathbf{p}_n^{i'}\right]^{\mathrm{T}} \tag{3.44}$$

$$\mathbf{U}_{i-1} = \left[\mathbf{p}_1^{i-1'} \quad \mathbf{p}_2^{i-1'} \quad \cdots \quad \mathbf{p}_i^{i-1'} \quad \cdots \quad \mathbf{p}_{n-1}^{i-1'} \quad \mathbf{p}_n^{i-1'}\right]^{\mathrm{T}} \tag{3.45}$$

Stopping criteria of optimization is set as Eq.3.46 [13]. When $res < tol$, the optimization will stop. tol is set to 0.001 in this paper. $\max(\mathbf{dU}_i)$ means the max value of $\mathbf{dU}_i$. The definitions of $\mathbf{dU}_i$ and $\mathbf{dUV}$ are as Eq.3.47 and 3.48.

$$res = \frac{\max(\mathbf{dU}_i)}{\max(\mathbf{dUV})} \tag{3.46}$$

$$\mathbf{dU}_i = \left[||\mathbf{p}_1^{i'} - \mathbf{p}_1^{i-1'}||^2 \quad \cdots \quad ||\mathbf{p}_i^{i'} - \mathbf{p}_i^{i-1'}||^2 \quad \cdots \quad ||\mathbf{p}_n^{i'} - \mathbf{p}_n^{i-1'}||^2 \right]^{\mathrm{T}} \tag{3.47}$$

$$\mathbf{dUV} = \left[||\mathbf{p}_1^{i'} - \mathbf{p}_1||^2 \quad \cdots \quad ||\mathbf{p}_i^{i'} - \mathbf{p}_i||^2 \quad \cdots \quad ||\mathbf{p}_n^{i'} - \mathbf{p}_n||^2 \right]^{\mathrm{T}} \tag{3.48}$$

3.6 Results and Discussion

In this section, we will show the deformation results performed by this optimization method, and analyze reasons of the existing probelms.

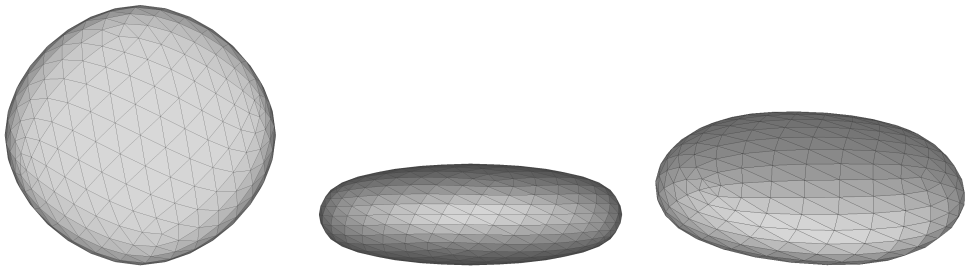
3.6.1 Results

In Fig.3.1, input mesh model is like a go stone, and Fig.3.1(a)-3.1(c) show the input model from three different views. The number of the input model is 386. The deformation results with $\lambda = 1$ are shown in Fig.3.2. Figure 3.2 and 3.1 are in same views. The surface center become a little concave after deformation shown as Fig.3.2(c). At the bottom, the surface center is also a little concave. When enlarge λ, the deformation results are almost no difference except the change in scale. λ is larger, the deformation shape of the go stone model is bigger.

The energy changing process is shown in Fig.3.3. After every iteration of optimization, the energy values are recorded, and shown in Fig.3.3(a). In Fig.3.3(a), the vertical axis represents the energy value, and the horizontal axis represents the number of iterations. Figure 3.3(a) shows energy value after each iteration. After optimization, the energy is reduced, indicating that the optimization method is effective.

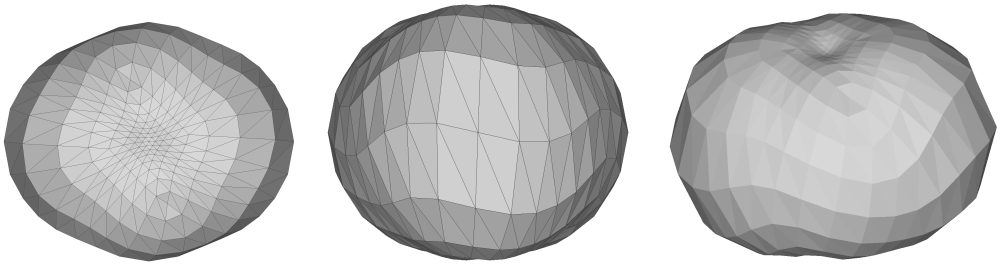
Global energy changing records the energy difference after and before global step optimization of every iteration and shown in Fig.3.3(b). The

vertical axis represents the value of energy difference, and the horizontal axis represents the number of iterations. The energy difference is less than 0, indicating that the energy is reduced after global step optimization.



(a) front view (b) top view (c) side view

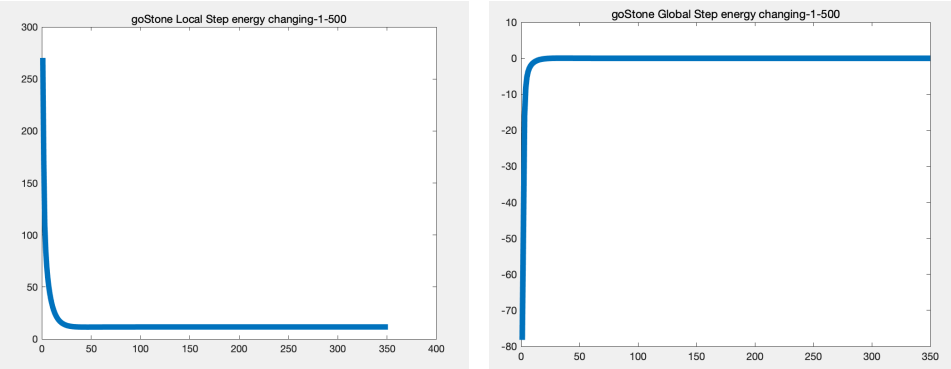
Figure 3.1: Input model of goStone



(a) front view (b) top view (c) side view

Figure 3.2: Deformation of goStone with $\lambda = 1$

However, energy does not guarantee convergence for all input models. A simplified stone house model is performed on this optimization method. But the energy can not converge shown in Fig.3.5(a). In Fig.3.5(a), the energy increased after optimization. Also, after global step optimization, the phenomenon of energy increase will appear shown in Fig.3.5(b). When the energy can not converge, deformation is ineffective. Original stone house model (© Perry Engel under CC BY) is from thingi10K.



(a) local energy change (b) global energy change

Figure 3.3: Energy Changing of goStone

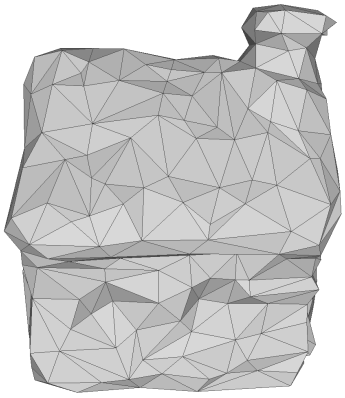
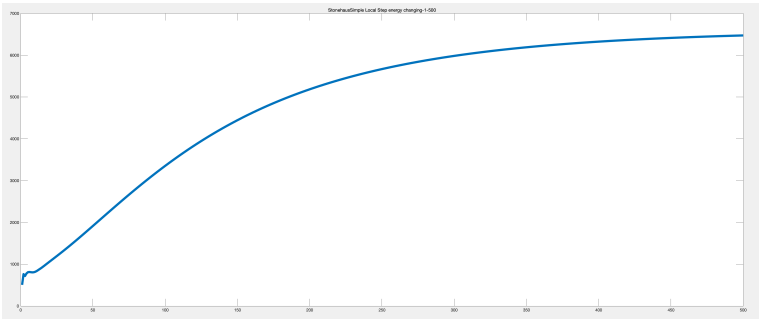


Figure 3.4: Input model of simplified stone house

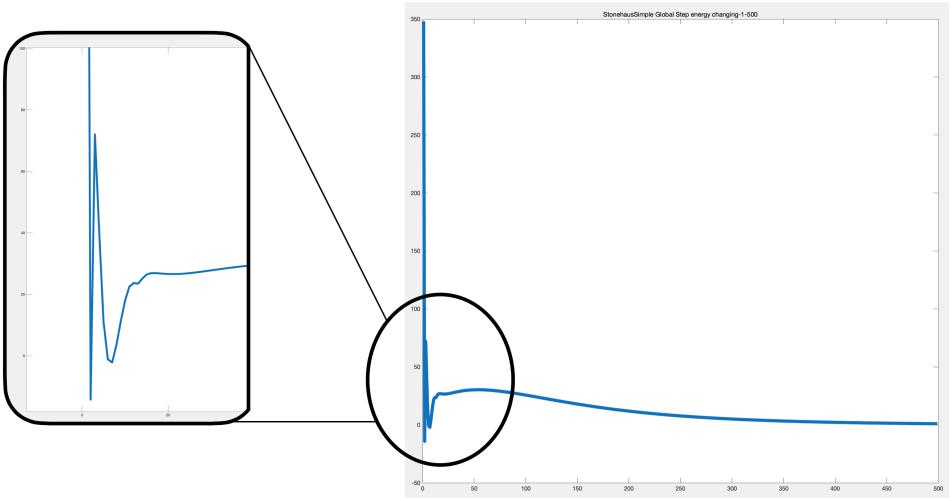
3.6.2 Discussion

The optimization solution of nonlinear spherical feature has problems as follows.

(1) The matrix $\mathbf{L}_{non2}$ is changed in every iteration, due to auxiliary variable s_i. After optimization of s_i in every iteration, $\mathbf{L}_{non2}$ will change. Therefore, $\mathbf{L}_{non2}$ can not be pre-factored. In every iteration, LU factorization is performed on $\mathbf{L}_{non2}$. This results in very slow optimization. When the point number of the input model is larger than one thousand, the algorithm performs too much time.



(a) local energy change of simplified stonehouse



(b) global energy change of simplified stonehouse

Figure 3.5: Energy Changing of simplified stonehouse

(2) At the same time, the matrix $\mathbf{L}_{non2}$ is affected due to the influence of the value s_i. In the optimization method, in order to avoid different orders of magnitude between W_i and g_i, we unitize the scale of the iput mesh model. Even though, energy is not guaranteed to converge when performing the optimization method on input mesh models.

(3) When optimizing via auxiliary variables s_i, the radius is not explicitly constrained in this optimization method. When the normal vector is parallel to the vector from the center to the point, there are possibilities that the point may become slightly concave after deformation.

3.7 Summary

In this chapter, we show the optimization process about combination of ARAP and nonlinear spherical feature. Nonlinear spherical feature makes optimization difficult. Although we introduce auxiliary variables for optimization. But after introducing auxiliary variables, the matrix changes after each iteration, and energy convergence cannot be guaranteed.

It is difficult to solve nonlinear spherical feature energy optimization. The method of introducing auxiliary variables did not achieve a successful optimization solution. Next we will explore a linear spherical feature and avoid these problems.

Chapter 4

Linear Spherical Feature and Optimization Process

4.1 Overview

In this chapter, we explore a proper spherical feature and describe the feature as ℓ_2-regularization (linear form). Based on this linear spherical feature, the optimization of this algorithm, combining ARAP and the linear spherical feature, is solved. The process of the optimization is introduced in details. To verify the convergence and deformation effectiveness, simple and complex models are performed by this algorithm.

4.2 Linear Spherical Feature

Based on the above knowledge, we propose an easier and efficient method to describe spherical feature. The feature of the sphere shape is as Fig.4.1. The unit normal vector difference between any two points on the spherical surface is proportional to the corresponding position vector difference and this ratio is equal to the radius of the sphere. In Fig.4.1, $\mathbf{p}_1$ and $\mathbf{p}_2$ are positions of two

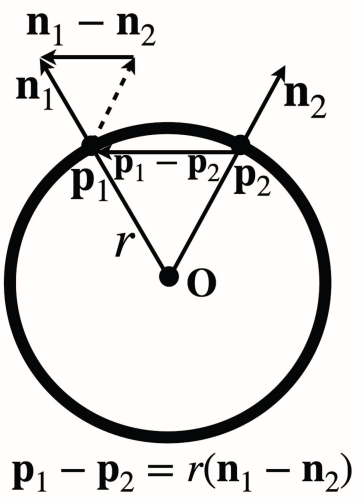


Figure 4.1: Spherical feature

points on the spherical surface. $\mathbf{n}_1$ and $\mathbf{n}_2$ are unit normal vectors of point $\mathbf{p}_1$ and $\mathbf{p}_2$ respectively. r is the radius of the sphere. $\mathbf{O}$ is the sphere center. According to the spherical feature, the two points satisfy Eq.4.1.

$$\mathbf{p}_1 - \mathbf{p}_2 = r(\mathbf{n}_1 - \mathbf{n}_2) \tag{4.1}$$

There is linear relationship with respect to $\mathbf{p}_1$ and $\mathbf{p}_2$ in Eq.4.1. According to this linear feature, we construct the energy function as Eq.4.2.

$$\begin{aligned} \sum_{i \in \mathcal{V}} \sum_{(i,j) \in \varepsilon_i} w_{ij} ||(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 + \\ \lambda \sum_{(i,j) \in \varepsilon_i} ||r\mathbf{R}_i(\mathbf{n}_i - \mathbf{n}_j) - (\mathbf{p}'_i - \mathbf{p}'_j)||^2 \end{aligned} \tag{4.2}$$

In Eq.4.2, the first term is the ARAP energy [20]. The second term is the spherical feature energy, and it is ℓ_2-regularization. In Eq.4.2, with the ARAP and spherical terms, the optimal energy can preserve the local structure and encourage the model with the spherical feature, such as rounder surface. λ is a controlling parameter to balance the ARAP term and spherical term. w_{ij} is the cotangent weight [16, 18]; ε_i is the "spokes and rims" edges of the ith point [3]; $\mathbf{p}'_i$ and $\mathbf{p}_i$ are the deformed and original ith point positions respectively;

$\mathbf{R}_i$ is the rotation matrix; $\mathbf{n}_i$ is the unit normal vector of the ith point in the original input model. ARAP term and spherical feature term both are linear with respect to $\mathbf{p}'_i$. The linear relationship makes optimization easier. We assign $r = \sqrt{A/4\pi}$, and A is the total area of the original input surface.

Observing Fig.4.1, we can also introduce other linear forms. The vector from sphere center to spherical surface point is as a position direction. Position directions and position difference also satisfy the same form, that is, replace $\mathbf{n}_1$ and $\mathbf{n}_2$ with position direction $\mathbf{d}_1$ and $\mathbf{d}_2$. Equation 4.3 is also the spherical feature.

$$\mathbf{p}_1 - \mathbf{p}_2 = r(\mathbf{d}_1 - \mathbf{d}_2) \tag{4.3}$$

The definition of $\mathbf{d}_i$ is as follows. Connect sphere center $\mathbf{O}$ and each point $\mathbf{p}_i$ of the spherical surface. Then we can get a vector from $\mathbf{O}$ to each point $\mathbf{p}_i$. Normalize the vector as position direction $\mathbf{d}_i$ shown as Eq.4.4.

$$\mathbf{d}_i = \frac{\mathbf{p}_i - \mathbf{O}}{||\mathbf{p}_i - \mathbf{O}||} \qquad (i \in \mathcal{V}) \tag{4.4}$$

Suppose point $\mathbf{p}_i$ and point $\mathbf{p}_j$ are on spherical surface. The linear form in Eq.4.5 also satisfies the spherical feature.

$$\mathbf{p}_i - \mathbf{p}_j = r(\mathbf{d}_i - \mathbf{d}_j) \tag{4.5}$$

Interpolate the normal vector and the position direction shown as Eq.4.6. $\mathbf{t}_i$ is the target direction of the ith point, which is interpolated by position direction $\mathbf{d}_i$ and the normal vector $\mathbf{n}_i$.

$$\mathbf{t}_i = (1 - a) * \mathbf{d}_i + a * \mathbf{n}_i \qquad (0 \leq a \leq 1) \tag{4.6}$$

Then a cluster of linear forms can be obtained shown as Eq.4.7. These linear forms also satisfy the spherical features.

$$\mathbf{p}_i - \mathbf{p}_j = r(\mathbf{t}_i - \mathbf{t}_j) \tag{4.7}$$

Based on above linear spherical features, the energy function, combing ARAP and spherical feature, is as Eq.4.8.

$$\sum_{i\in\mathcal{V}}\sum_{(i,j)\in\varepsilon_i} w_{ij}||(\mathbf{p}'_i-\mathbf{p}'_j)-\mathbf{R}_i(\mathbf{p}_i-\mathbf{p}_j)||^2+ \\ \lambda\sum_{(i,j)\in\varepsilon_i}||r\mathbf{R}_i(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)||^2 \tag{4.8}$$

Minimize the energy function as the optimization target. Because of the linear form, it is easy to optimize. We adopt local-global update strategy to optimize the energy function, same as the optimization method of ARAP surface modeling [20]. In local step, optimize the rotation matrices $\{\mathbf{R}_i\}$; In global step, optimize point positions $\mathbf{V}'$.

4.2.1 Parameters

In Eq.4.6, a is an interpolation parameter, and determines target directions. The domain of a is between 0 and 1. Assuming λ is very large, when a is equal to 0 (only position directions as target directions), the deformed model looks globally rounder. When a is equal to 1 (only normal vectors as target direcitons), the curved corners of the input model are obviously deformed rounder. When the value of a is between 0 and 1, the deformation effectiveness is in between.

In Eq.4.8, λ is a weight value to balance the ARAP term and spherical term. The domain of λ is larger than 0. λ is larger, and the effect of spherical style is more. According to our experimental experience, $\lambda = 5$ is a very large weight value.

Sphere center $\mathbf{O}$ is also a parameter. We set $\mathbf{O}$ as the mean coordinate value of all points of the input model by default. Users can set the coordinates of $\mathbf{O}$. The closer the surface mesh of the input model to $\mathbf{O}$, the rounder the surface can be deformed. If $\mathbf{O}$ is outside the input model, the surface can be

deformed concavely rounder.

User can try different values of λ, a and $\mathbf{O}$. In experimental results of this chapter, we investigate and discuss the effectiveness and influence of the interpolation linear form as spherical feature and parameter λ.

4.3 Local Step Optimization

The purpose of local step optimization is to solve the optimal rigid transformation matrix $\{\mathbf{R}_i\}$, under the premise of known point positions $\{\mathbf{p}_i\}$, $\{\mathbf{p}'_i\}$.

Given the cell C_i, which covers the triangles incident upon the ith (refering to Fig.2.3), and its deformed cell C'_i [20], the energy related to C_i and C'_i is as 4.9:

$$\begin{aligned} E(C_i, C'_i) = & \sum_{(i,j)\in\varepsilon_i} w_{ij} ||(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 \\ & + \lambda \sum_{(i,j)\in\varepsilon_i} ||r\mathbf{R}_i(\mathbf{t}_i - \mathbf{t}_j) - (\mathbf{p}'_i - \mathbf{p}'_j)||^2 \end{aligned} \tag{4.9}$$

Minimize the energy of $E(C_i, C'_i)$, and expand Eq.4.9 as Eq.4.10. $\mathbf{e}_{ij} = \mathbf{p}_i - \mathbf{p}_j$.

$$\begin{aligned} & \underset{\mathbf{R}_i}{\text{minimize}} \quad E(C_i, C'_i) \\ = & \underset{\mathbf{R}_i}{\text{minimize}} \sum_{(i,j)\in\varepsilon_i} w_{ij} ||(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 \\ & + \lambda \sum_{(i,j)\in\varepsilon_i} ||r\mathbf{R}_i(\mathbf{t}_i - \mathbf{t}_j) - (\mathbf{p}'_i - \mathbf{p}'_j)||^2 \\ = & \underset{\mathbf{R}_i}{\text{minimize}} \sum_{(i,j)\in\varepsilon_i} w_{ij} (\mathbf{e}'_{ij} - \mathbf{R}_i\mathbf{e}_{ij})^{\mathrm{T}} (\mathbf{e}'_{ij} - \mathbf{R}_i\mathbf{e}_{ij}) \\ & + \lambda \sum_{(i,j)\in\varepsilon_i} [r\mathbf{R}_i(\mathbf{t}_i - \mathbf{t}_j) - \mathbf{e}'_{ij}]^{\mathrm{T}} [r\mathbf{R}_i(\mathbf{t}_i - \mathbf{t}_j) - \mathbf{e}'_{ij}] \\ = & \underset{\mathbf{R}_i}{\text{minimize}} \sum_{(i,j)\in\varepsilon_i} w_{ij} ({\mathbf{e}'_{ij}}^{\mathrm{T}} \mathbf{e}'_{ij} - 2{\mathbf{e}_{ij}}^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{e}'_{ij} + {\mathbf{e}_{ij}}^{\mathrm{T}} \mathbf{e}_{ij}) \\ & + \lambda \sum_{(i,j)\in\varepsilon_i} [r^2(\mathbf{t}_i - \mathbf{t}_j)^{\mathrm{T}} (\mathbf{t}_i - \mathbf{t}_j) - 2r(\mathbf{t}_i - \mathbf{t}_j)^{\mathrm{T}} {\mathbf{R}_i}^{\mathrm{T}} \mathbf{e}'_{ij} + {\mathbf{e}'_{ij}}^{\mathrm{T}} \mathbf{e}'_{ij}] \end{aligned} \tag{4.10}$$

Drop constant terms, the minimization is as Eq.4.11:

$$\underset{\mathbf{R}_i}{\text{minimize}} \sum_{(i,j)\in\varepsilon_i} -2w_{ij}{\mathbf{e}'_{ij}}^{\mathrm{T}}\mathbf{R}_i\mathbf{e}_{ij} + \lambda \sum_{(i,j)\in\varepsilon_i} -2r{\mathbf{e}'_{ij}}^{\mathrm{T}}\mathbf{R}_i(\mathbf{t}_i - \mathbf{t}_j) \tag{4.11}$$

Equation 4.11 is equivalent to Eq.4.12.

$$\begin{aligned} &\arg\max_{\mathbf{R}_i} \sum_{(i,j)\in\varepsilon_i} w_{ij}{\mathbf{e}'_{ij}}^{\mathrm{T}}\mathbf{R}_i\mathbf{e}_{ij} + \lambda \sum_{(i,j)\in\varepsilon_i} r{\mathbf{e}'_{ij}}^{\mathrm{T}}\mathbf{R}_i(\mathbf{t}_i - \mathbf{t}_j) \\ =&\arg\max_{\mathbf{R}_i} \ \mathrm{tr}(\mathbf{R_i}\mathbf{S_i}) \end{aligned} \tag{4.12}$$

$\mathbf{S}_i$ is as Eq.4.13:

$$\mathbf{S}_i = \sum_{(i,j)\in\varepsilon_i} [w_{ij}\mathbf{e}_{ij}{\mathbf{e}'_{ij}}^{\mathrm{T}} + \lambda r(\mathbf{t}_i - \mathbf{t}_j){\mathbf{e}'_{ij}}^{\mathrm{T}}] \tag{4.13}$$

One can derive $\mathbf{R}_i$ from the singular value decomposition(SVD) as Eq.4.14.

$$\mathbf{S}_i = \mathbf{U}_i \sum\nolimits_i \mathbf{Q}_i^{\mathrm{T}} \tag{4.14}$$

$\mathbf{R}_i$ is as Eq.4.15.

$$\mathbf{R}_i = \mathbf{Q}_i\mathbf{U}_i^{\mathrm{T}} \tag{4.15}$$

In Eq.4.15, if $det(\mathbf{R}_i) < 0$, the last column of $\mathbf{U}_i$ needs to be negated.

4.4 Global Step Optimization

The energy function of the whole mesh is to add the energy of per cell $E(C_i, C'_i)$, expressed as Eq.4.16. The number of points is n.

$$\begin{aligned} E(M') =& \sum_{i=1}^{n} E(C_i, C'_i) \\ =& \sum_{i=1}^{n} \sum_{(i,j)\in\varepsilon_i} w_{ij}||(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 \\ &+ \lambda \sum_{(i,j)\in\varepsilon_i} ||r\mathbf{R}_i(\mathbf{t}_i - \mathbf{t}_j) - (\mathbf{p}'_i - \mathbf{p}'_j)||^2 \end{aligned} \tag{4.16}$$

$E(M')$ depends only on the geometries of the original model, and the deformed model, i.e., on the point positions $\mathbf{p}$, and $\mathbf{p}'$.

Minimize the total energy (shown as Eq.4.16) as the optimization target of global step. The purpose of global step optimization is to solve the optimal point positions $\{\mathbf{p}'_i\}$ under the premise of known transformation matrix $\{\mathbf{R}_i\}$.

In Eq.4.16, the function $E(M')$ is a quadratic function with respect to the point positions $\{\mathbf{p}'_i\}$. To calculate the optimal point positions $\{\mathbf{p}'_i\}$, calculate the partial derivative of $E(M')$ about each point position $\mathbf{p}'$. For the convenience of writing, $E(M')$ in Eq.4.16 is divided into two parts E_1 and E_2 shown as Eq.4.17. The partial derivative of E_1 about $\mathbf{p}'$ is as Eq.4.18. The partial derivative of E_2 about $\mathbf{p}'$ is as Eq.4.19.

We adopt spoke-and-rims structure shown as Fig.4.2. $N(i)$ is one ring neighbors of the ith point, not including the ith point. $\mathbf{R}_m$ and $\mathbf{R}_n$ are rotation matrices of the mth and nth points, which are the opposite points $\mathbf{p}_m$ and $\mathbf{p}_n$ of the common edge vector $\mathbf{e}_{ij}$ in triangle face (i, n, j) and triangle face (i, j, m) [9].

$$\begin{cases} E(M') = E_1 + E_2 \\ E_1 = \sum_{i=1}^{n} \sum_{(i,j)\in\varepsilon_i} w_{ij} ||(\mathbf{p}'_i - \mathbf{p}'_j) - \mathbf{R}_i(\mathbf{p}_i - \mathbf{p}_j)||^2 \\ E_2 = \sum_{i=1}^{n} \sum_{(i,j)\in\varepsilon_i} \lambda ||r\mathbf{R}_i(\mathbf{t}_i - \mathbf{t}_j) - (\mathbf{p}'_i - \mathbf{p}'_j)||^2 \end{cases} \tag{4.17}$$

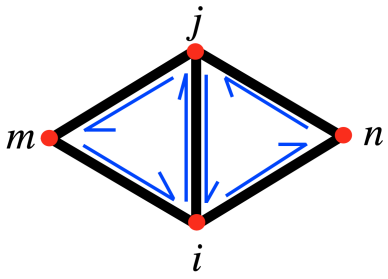


Figure 4.2: Spokes-and-rims applied in global step optimization

$$
\begin{aligned}
&\frac{\partial E_1}{\partial \mathbf{p}'_i} \\
=&\frac{\partial}{\partial \mathbf{p}'_i}\{\sum_{j\in N(i)}[w_{ij}||(\mathbf{p}'_i-\mathbf{p}'_j)-\mathbf{R}_i(\mathbf{p}_i-\mathbf{p}_j)||^2+w_{ji}||(\mathbf{p}'_j-\mathbf{p}'_i)-\mathbf{R}_j(\mathbf{p}_j-\mathbf{p}_i)||^2 \\
&+w_{ij}||(\mathbf{p}'_i-\mathbf{p}'_j)-\mathbf{R}_m(\mathbf{p}_i-\mathbf{p}_j)||^2+w_{ij}||(\mathbf{p}'_i-\mathbf{p}'_j)-\mathbf{R}_i(\mathbf{p}_i-\mathbf{p}_j)||^2 \\
&+w_{ji}||(\mathbf{p}'_j-\mathbf{p}'_i)-\mathbf{R}_j(\mathbf{p}_j-\mathbf{p}_i)||^2+w_{ji}||(\mathbf{p}'_j-\mathbf{p}'_i)-\mathbf{R}_n(\mathbf{p}_j-\mathbf{p}_i)||^2]\} \\
=&\sum_{j\in N(i)}\{2w_{ij}[3(\mathbf{p}'_i-\mathbf{p}'_j)-(\mathbf{R}_i+\mathbf{R}_j+\mathbf{R}_m)(\mathbf{p}_i-\mathbf{p}_j)] \\
&+2w_{ji}[3(\mathbf{p}'_i-\mathbf{p}'_j)-(\mathbf{R}_i+\mathbf{R}_j+\mathbf{R}_n)(\mathbf{p}_i-\mathbf{p}_j)]\} \\
=&\sum_{j\in N(i)}2w_{ij}[6(\mathbf{p}'_i-\mathbf{p}'_j)-(2\mathbf{R}_i+2\mathbf{R}_j+\mathbf{R}_m+\mathbf{R}_n)(\mathbf{p}_i-\mathbf{p}_j)]
\end{aligned}
\tag{4.18}
$$

$$
\begin{aligned}
&\frac{\partial E_2}{\partial \mathbf{p}'_i} \\
=&\frac{\partial}{\partial \mathbf{p}'_i}\lambda\sum_{j\in N(i)}\{||r\mathbf{R}_i(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)||^2+||r\mathbf{R}_j(\mathbf{t}_j-\mathbf{t}_i)-(\mathbf{p}'_j-\mathbf{p}'_i)||^2 \\
&+||r\mathbf{R}_m(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)||^2+||r\mathbf{R}_i(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)||^2 \\
&+||r\mathbf{R}_j(\mathbf{t}_j-\mathbf{t}_i)-(\mathbf{p}'_j-\mathbf{p}'_i)||^2+||r\mathbf{R}_n(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)||^2\} \\
=&\lambda\sum_{j\in N(i)}\{-2[r\mathbf{R}_i(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)]+2[r\mathbf{R}_j(\mathbf{t}_j-\mathbf{t}_i)-(\mathbf{p}'_j-\mathbf{p}'_i)] \\
&-2[r\mathbf{R}_m(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)]-2[r\mathbf{R}_i(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)] \\
&+2[r\mathbf{R}_j(\mathbf{t}_j-\mathbf{t}_i)-(\mathbf{p}'_j-\mathbf{p}'_i)]-2[r\mathbf{R}_n(\mathbf{t}_i-\mathbf{t}_j)-(\mathbf{p}'_i-\mathbf{p}'_j)]\} \\
=&2\lambda\sum_{j\in N(i)}[6(\mathbf{p}'_i-\mathbf{p}'_j)-r(2\mathbf{R}_i+2\mathbf{R}_j+\mathbf{R}_m+\mathbf{R}_n)(\mathbf{t}_i-\mathbf{t}_j)]
\end{aligned}
\tag{4.19}
$$

Add Eq.4.18 and 4.19 as the the partial derivative of $E(M')$ about $\mathbf{p}'_i$ shown

as Eq.4.20.

$$\begin{aligned}\frac{\partial E(M')}{\partial \mathbf{p}'_i} =& \frac{\partial E_1}{\partial \mathbf{p}'_i} + \frac{\partial E_2}{\partial \mathbf{p}'_i} \\ =& \sum_{j \in N(i)} 2w_{ij}[6(\mathbf{p}'_i - \mathbf{p}'_j) - (2\mathbf{R}_i + 2\mathbf{R}_j + \mathbf{R}_m + \mathbf{R}_n)(\mathbf{p}_i - \mathbf{p}_j)] \\ &+ 2\lambda[6(\mathbf{p}'_i - \mathbf{p}'_j) - r(2\mathbf{R}_i + 2\mathbf{R}_j + \mathbf{R}_m + \mathbf{R}_n)(\mathbf{t}_i - \mathbf{t}_j)]\end{aligned} \tag{4.20}$$

Set the partial derivatives to zero w.r.t. each $\mathbf{p}'_i$, such that set $\frac{\partial E(M')}{\partial \mathbf{p}'_i} = 0$ shown as Eq.4.21.

$$\begin{aligned}&\sum_{j \in N(i)} [2w_{ij} \times 6(\mathbf{p}'_i - \mathbf{p}'_j) + 2\lambda \times 6(\mathbf{p}'_i - \mathbf{p}'_j)] \\ =& \sum_{j \in N(i)} [2w_{ij}(2\mathbf{R}_i + 2\mathbf{R}_j + \mathbf{R}_m + \mathbf{R}_n)(\mathbf{p}_i - \mathbf{p}_j) + 2\lambda r(2\mathbf{R}_i + 2\mathbf{R}_j + \mathbf{R}_m + \mathbf{R}_n)(\mathbf{t}_i - \mathbf{t}_j)]\end{aligned} \tag{4.21}$$

In Eq.4.21, both sides of the equal sign are reduced by a factor of 12 at the same time. Then we get Eq.4.22.

$$\begin{aligned}&\sum_{j \in N(i)} (w_{ij} + \lambda)(\mathbf{p}'_i - \mathbf{p}'_j) \\ =& \frac{1}{6} \sum_{j \in N(i)} [w_{ij}(2\mathbf{R}_i + 2\mathbf{R}_j + \mathbf{R}_m + \mathbf{R}_n)(\mathbf{p}_i - \mathbf{p}_j) \\ &+ \lambda r(2\mathbf{R}_i + 2\mathbf{R}_j + \mathbf{R}_m + \mathbf{R}_n)(\mathbf{t}_i - \mathbf{t}_j)]\end{aligned} \tag{4.22}$$

The partial derivative with respect to each point $\mathbf{p}'_\mathbf{i}, i \in \mathcal{V}$ corresponds to an equation. All equations can be organized into matrix form as Eq.4.23.

$$\mathbf{L_0} \times \mathbf{P}'_0 = \mathbf{B}_0 \tag{4.23}$$

$\mathbf{P}'_0$ is as Eq.4.24. The size of $\mathbf{P}'_0$ is $n \times 3$. Point position $\mathbf{p}'_i$ is a 3×1 vector. In Eq.4.24, vector of $\mathbf{p}'_i$ is transposed by default.

$$\mathbf{P}'_0 = \begin{bmatrix}\mathbf{p}'_1 & \mathbf{p}'_2 & \cdots & \mathbf{p}'_i & \cdots & \mathbf{p}'_n\end{bmatrix}^{\mathrm{T}} \tag{4.24}$$

$\mathbf{B}_0$ is as Eq.4.25. The size of $\mathbf{B}_0$ is $n \times 3$.

$$\mathbf{B}_0 = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_i & \cdots & \mathbf{b}_n \end{bmatrix}^{\mathrm{T}} \tag{4.25}$$

$\mathbf{b}_i$ is a 1×3 vector shown as Eq.4.26.

$$\begin{aligned} \mathbf{b}_i = & \frac{1}{6} \sum_{j \in N(i)} [w_{ij}(2\mathbf{R}_i + 2\mathbf{R}_j + \mathbf{R}_m + \mathbf{R}_n)(\mathbf{p}_i - \mathbf{p}_j) \\ & + \lambda r(2\mathbf{R}_i + 2\mathbf{R}_j + \mathbf{R}_m + \mathbf{R}_n)(\mathbf{t}_i - \mathbf{t}_j)]^{\mathrm{T}} \end{aligned} \tag{4.26}$$

The matrix $\mathbf{L}_0$ is the coefficients of all equations. The size of $\mathbf{L}_0$ is $n \times n$. The definition of $\mathbf{L}_0$ is as Alg.1. The matrix form of $\mathbf{L}_0$ is shown as Eq.4.27.

Algorithm 1 The definition of $\mathbf{L}_0$

```
for each i ∈ V do
  for each j ∈ V do
    if j == i then
      L_0(i, i) ← Σ_{j∈N(i)} (w_ij + λ)
    else
      if j ∈ N(i) then
        L_0(i, j) ← (−w_ij − λ)
      else
        L_0(i, j) ← 0
      end if
    end if
  end for
end for
```

$$\begin{bmatrix} \sum_{j\in N(1)}(w_{1j}+\lambda) & \cdots & -w_{1i}-\lambda & \cdots & -w_{1,n-1}-\lambda & -w_{1,n}-\lambda \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{i,1}-\lambda & \cdots & \sum_{j\in N(i)}(w_{ij}+\lambda) & \cdots & -w_{i,n-1}-\lambda & -w_{i,n}-\lambda \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{n-1,1}-\lambda & \cdots & -w_{n-1,i}-\lambda & \cdots & \sum_{j\in N(n-1)}(w_{n-1,j}+\lambda) & -w_{n-1,n}-\lambda \\ -w_{n,1}-\lambda & \cdots & -w_{n,i}-\lambda & \cdots & -w_{n,n-1}-\lambda & \sum_{j\in N(n)}(w_{nj}+\lambda) \end{bmatrix} \tag{4.27}$$

The matrix $\mathbf{L}_0$ is sparse, symetric and not full rank. About not full rank, we can consider from the following view. If we obtain all points' positions of the input model after deformation, then we rotate the deformed model, and after rotation the model can be also one option that satisfies the spherical style deformation. Therefore, the matrix $\mathbf{L}_0$ is not full rank, and the solution is not unique.

Because the matix is not full rank, it is necessary to fix points to obtain unique solution. One rotation $\mathbf{R}$ has three free degrees. One point corresponds three equations. At least fix one point to achieve solution. We can fix any one point position. To convenience, we fix the point position $\mathbf{p}_n$. The solver process of other $\{\mathbf{p}'_i\}$ are similar to section 3.4.2. The solver process with fixed point $\mathbf{p}_n$ is as Eq.4.28.

$$\mathbf{L}_0 \times (\mathbf{P}'_{var} + \mathbf{P}'_{fix}) = \mathbf{B}_0 \tag{4.28}$$

$\mathbf{P}'_{var}$ and $\mathbf{P}'_{fix}$ satisfy the relationship as Eq.4.29. $\mathbf{o_{cst}} = [0 \quad 0 \quad 0]$.

$$\begin{aligned}\mathbf{P}'_0 =& \mathbf{P}'_{var} + \mathbf{P}'_{fix} \\ =& \begin{bmatrix} \mathbf{p}'_1 \\ \mathbf{p}'_2 \\ \vdots \\ \mathbf{p}'_i \\ \vdots \\ \mathbf{p}'_{n-1} \\ \mathbf{o}_{cst} \end{bmatrix} + \begin{bmatrix} \mathbf{o}_{cst} \\ \mathbf{o}_{cst} \\ \vdots \\ \mathbf{o}_{cst} \\ \vdots \\ \mathbf{o}_{cst} \\ \mathbf{p}_n \end{bmatrix}\end{aligned} \tag{4.29}$$

Equation 4.28 transforms to Eq.4.30.

$$\mathbf{L}_0 \times \mathbf{P}'_{var} = \mathbf{B}_0 - \mathbf{L}_0 \times \mathbf{P}'_{fix} \tag{4.30}$$

Because of fixed point $\mathbf{p}'_n$, now the number of variables reduces into $(n-1)$. To solve these variables, Eq.4.30 needs further simplification. Delete the coefficients corresponding to $\mathbf{o}_{cst}$ in matrix $\mathbf{L}_0$, that is, delete the nth column and nth row of $\mathbf{L}_0$. At the same time, delete the nth row of $\mathbf{P}'_{var}$, and delete the nth row of $(\mathbf{B}_0 - \mathbf{L}_0 \times \mathbf{P}'_{fix})$. After above simplifications, Eq.4.30 transforms into Eq.4.31. The size of $\mathbf{L}$ is $(n-1) \times (n-1)$. The size of $\mathbf{P}'_1$ is $(n-1) \times 3$. The size of $\mathbf{B}$ is $(n-1) \times 3$.

$$\mathbf{L} \times \mathbf{P}'_1 = \mathbf{B} \tag{4.31}$$

$\mathbf{P}'_1$ is as Eq.4.32.

$$\mathbf{P}'_1 = \begin{bmatrix} \mathbf{p}'_1 & \mathbf{p}'_2 & \cdots & \mathbf{p}'_i & \cdots & \mathbf{p}'_{n-1} \end{bmatrix}^{\mathrm{T}} \tag{4.32}$$

The matrix $\mathbf{L}$ is as Eq.4.33.

$$\begin{bmatrix} \sum_{j\in N(1)}(w_{1j}+\lambda) & \cdots & -w_{1i}-\lambda & \cdots & -w_{1,n-1}-\lambda \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{i,1}-\lambda & \cdots & \sum_{j\in N(i)}(w_{ij}+\lambda) & \cdots & -w_{i,n-1}-\lambda \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -w_{n-1,1}-\lambda & \cdots & -w_{n-1,i}-\lambda & \cdots & \sum_{j\in N(n-1)}(w_{n-1,j}+\lambda) \end{bmatrix} \tag{4.33}$$

The matrix $\mathbf{L}$ is still sparse, and symmetric. The importance is that in every iteration, $\mathbf{L}$ does not change. This means that the matrix $\mathbf{L}$ can be only pre-factored once for efficiency. Also, the pre-factored method is also LU factorization as Eq.4.34.

$$\mathbf{L} = \mathbf{lu} \tag{4.34}$$

After $\mathbf{L}$ is pre-factored, in every iteration, solving for $\mathbf{P}'_1$ only involves the calculation of Eq.4.37.

$$\begin{cases} \mathbf{l} \times \mathbf{X} = \mathbf{B} \\ \mathbf{X} = \mathbf{u} \times \mathbf{P}'_1 \end{cases} \tag{4.35}$$

$$\mathbf{X} = \mathbf{l}^{-1} \times \mathbf{B} \tag{4.36}$$

$$\mathbf{P}'_1 = \mathbf{u}^{-1} \times \mathbf{X} \tag{4.37}$$

Finally, the point positions $\mathbf{P}'$ is as Eq.4.38.

$$\mathbf{P}' = [\mathbf{P}'_1 \quad \mathbf{p}_n] \tag{4.38}$$

Optimization stopping criteria is same as section 3.5.

4.5 Results with Normal Direction

In this section, we performed the deformation method with $a = 1$ (only normal vectors as target directions) on three type models: convex and smooth models; convex and sharp models; complex models. When $a = 1$ (only normal vectors as target directions), the deformation method can deform the local curved surface rounder, and as the larger the roundness parameter λ, the rounder the local curved surface are deformed. At the same time, the problems of overly growing (and/or shrinking) deformation, and mesh self-intersections are found among deformation results. Then we show discussion about the two problems.

4.5.1 Deformation of Convex and Smooth Models

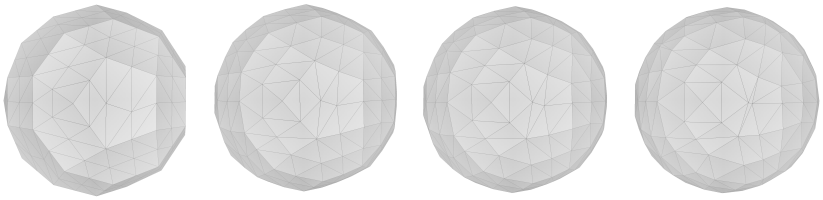
Firstly, we experimented on convex and smooth shapes. On convex and smooth shapes, our method can successfully converge. In these experiments, we give roundness parameter λ from 0.5 to 5. Sub-figures in the same row have the same view. The title "output with $\lambda = 0.5$" of sub-figures is shorted as "$\lambda = 0.5$", similarly as "$\lambda = 1$" and "$\lambda = 5$".

Model1 resembles a sphere shown in Fig.4.3(a). The larger the roundness parameter λ, the rounder the model is, shown in Fig.4.3(b)-4.3(d). Our method is able to deform the surface rounder while maintaining its topology.

About energy changing, the vertical axis represents the energy value, and the horizontal axis represents the number of iterations. The figure of local step energy records energy value after each iteration. The purpose of optimization is to obtain the minumum value of the energy. Therefore, after each iteration optimization, the energy is reduced until it converges to a value, indicating that the optimization method is effective. The figure of global step energy changing records the energy increment after and before global step optimiza-

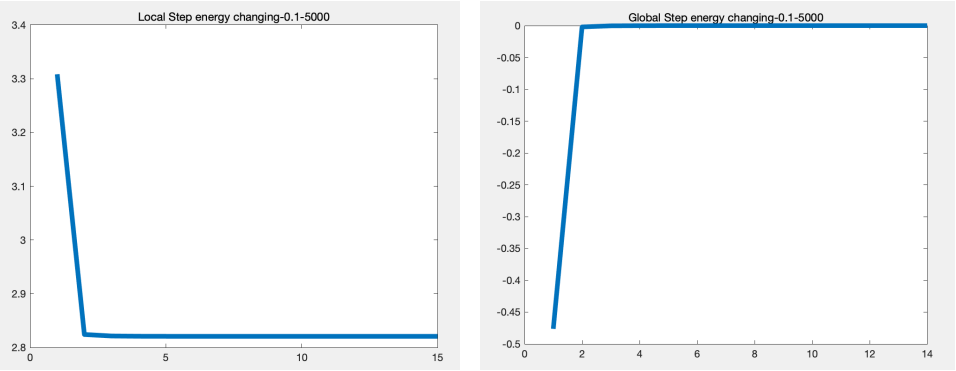
tion of each iteration. After every iteration of global step optimization, the energy increment is less than 0, indicating that the global step optimization is effective.

During the optimization process, the energy is always decreasing until it converges to a value shown in Fig.4.4(a). It also verifies that the global optimization is effective. Because after global optimization of every iteration, the energy increment is less than 0 shown in Fig.4.4(b).



(a) input (b) $\lambda = 0.5$ (c) $\lambda = 1$ (d) $\lambda = 5$

Figure 4.3: Deformation results of model1

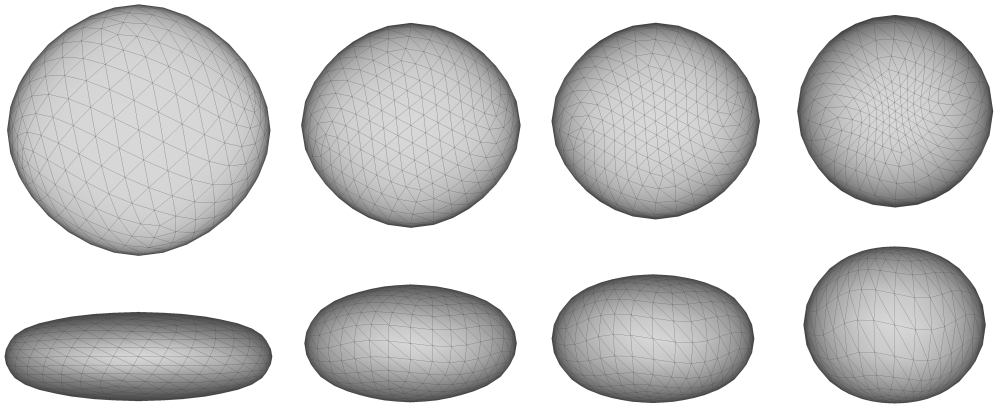


(a) local energy change (b) global energy change

Figure 4.4: Energy Changing of model1

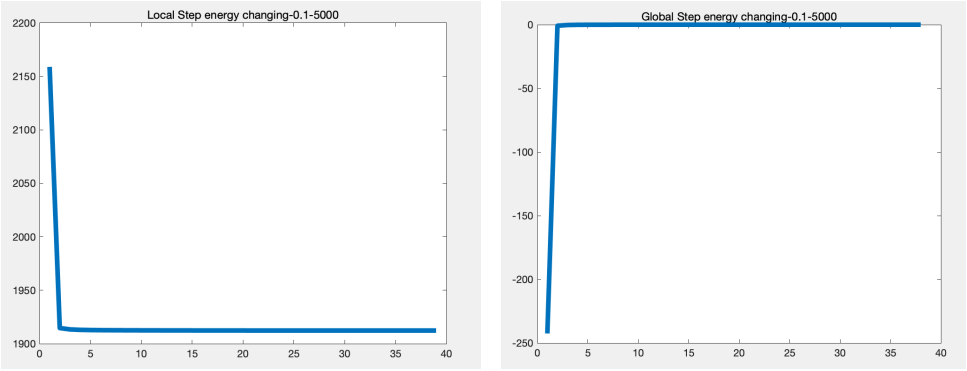
Model2 looks like a flat ellipsoid, like a go stone. In Fig.4.5, the first row shows from the top view and the second row shows from the side view. The larger the roundness parameter λ, again, the rounder the model is (Fig.4.5(b)-4.5(d)). In this model, the "side" parts of the model are growing and the "top

and bottom" parts are shrinking compared with the original model. Figure 4.6(a) and 4.6(b) show the energy optimization process is effective.



(a) input (b) $\lambda = 0.5$ (c) $\lambda = 1$ (d) $\lambda = 5$

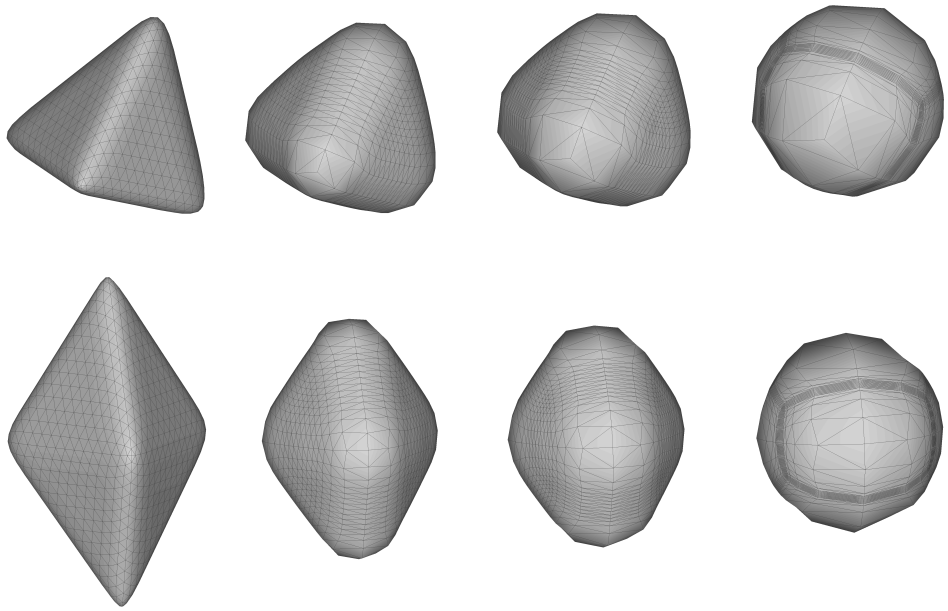
Figure 4.5: Deformation results of model2



(a) local energy change (b) global energy change

Figure 4.6: Energy Changing of model2

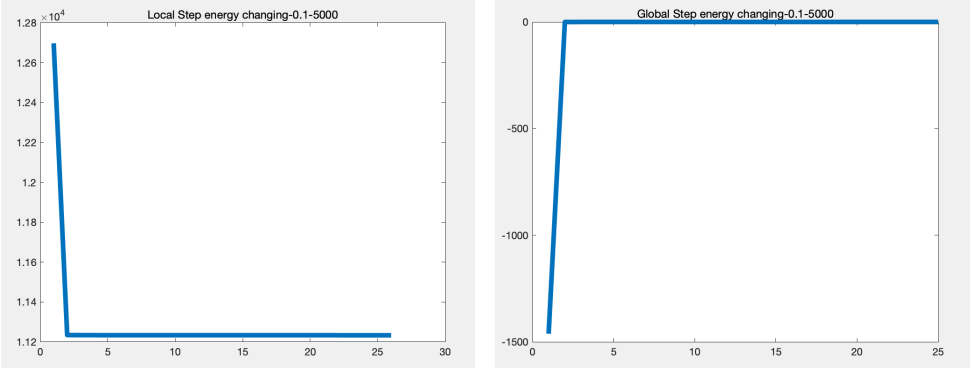
Model3 is like a shape of two regular tetrahedra connected together and it has several corners (Fig.4.7). In Fig.4.7, the first row shows from the side view and the second row shows from the front view. The larger the roundness parameter λ, again, the rounder the corners of the model are (Fig.4.7(b)-4.7(d)). At the same time, in this model, the corners are overly growing and the flat areas are overly shrinking with large λ. Also, the energy is converged, shown in Fig.4.8.



(a) input (b) $\lambda = 0.5$ (c) $\lambda = 1$ (d) $\lambda = 5$

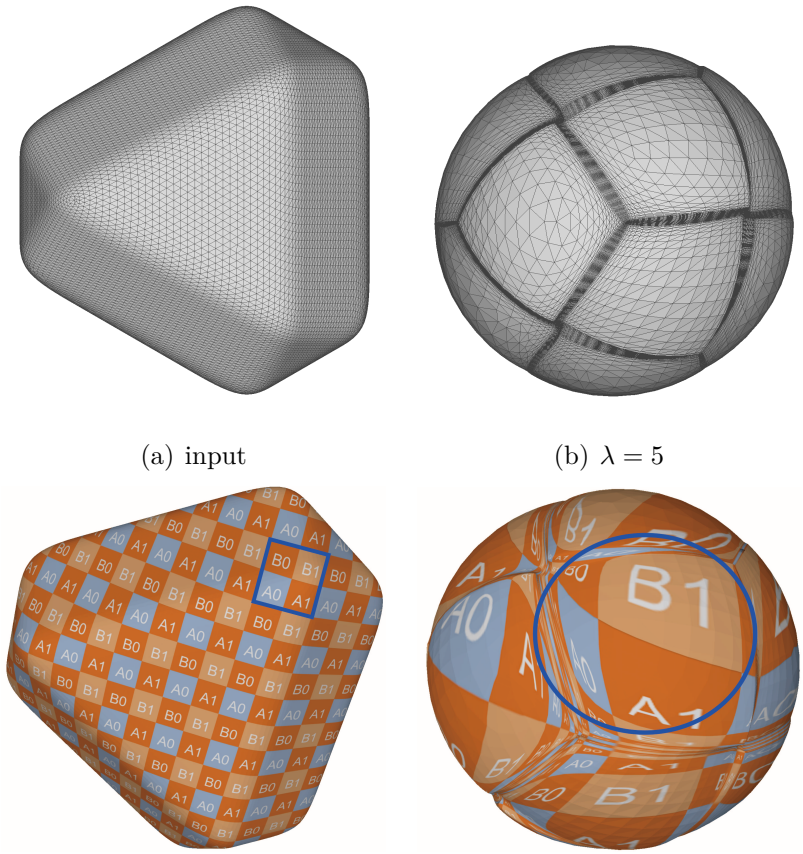
Figure 4.7: Deformation results of model3

To clearly show overly "growing" and "shrinking" deformation results, we deform another convex and smooth model4, namely, icosahedron shown as Fig.4.9(a). Figure 4.9(c) is the input model with texture. With $\lambda = 5$ and $a = 1$ (only normal vectors as target directions), the deformation result is shown as Fig.4.9(b), and Fig.4.9(d) is the deformation result with texture. In Fig.4.9(c), the area circled by the blue square is deformed into the area circled by the blue circle in Fig.4.9(d). The deformation method deforms the convex curved corner surface rounder. Also with texture information and large roundness parameter λ, the deformation results clearly show the overly growing and shrinking results. Figure 4.9(b) and 4.9(d) shows that "growing" occurs at the convex curved corner parts of the input shape and "shrinking" occurs at the flat parts, resulting in an uneven mesh density. Also, the deformation energy of icosahedron still converges.



(a) local energy change (b) global energy change

Figure 4.8: Energy Changing of model3



(a) input (b) $\lambda = 5$

(c) texture of (a) (d) texture of (b)

Figure 4.9: Deformation results of model4 (icosahedron)

4.5.1.1 Overly Growing and Shrinking

Figure 4.10 shows a discussion about the overly growing and shrinking problem, where the black ellipse represents the input shape "ellipsoid" and the orange circle represents the deformed shape "sphere" with large λ. Now due to large λ and simplicity of explanation, we ignore ARAP part. Each point of ellipsoid approximately deforms to the point on the sphere which has the approximately same normal vector as the point on ellipsoid. Normal vectors of points are represented as colorful pointed line segments. Point $\mathbf{p}_i$ of the input model approximately deforms to point $\mathbf{p}'_i$. The deformation may be explained as follows. In Eq.4.2, high curvature part of the input model ($\mathbf{p}_1$ and $\mathbf{p}_2$ in Fig.4.10) is expanded as a result of minimization because normal vector difference $\lambda||r(\mathbf{n}_i - \mathbf{n}_j)||$ is greater than position difference $\lambda||\mathbf{p}_i - \mathbf{p}_j||$. Oppositely, shrink occurs in flat areas ($\mathbf{p}_3$ and $\mathbf{p}_4$ in Fig.4.10).

Therefore, when $||r(\mathbf{n}_i - \mathbf{n}_j)||$ is larger than $||\mathbf{p}_i - \mathbf{p}_j||$, minimizing $||r\mathbf{R}_i(\mathbf{n}_i - \mathbf{n}_j) - (\mathbf{p}_i - \mathbf{p}_j)||$ deforms the area between $\mathbf{p}_i$ and $\mathbf{p}_j$ larger and overly growing with large λ. Oppositely, when $||r(\mathbf{n}_i - \mathbf{n}_j)||$ is smaller than $||\mathbf{p}_i - \mathbf{p}_j||$, this causes the area smaller and overly shrinking with large λ.

4.5.2 Deformation of Convex and Sharp Models

We try other convex models, which contain sharp edges and corners as Fig.4.11, 4.12, and 4.13.

Figure 4.11(a) is an input square model from the front view. Figure 4.11(b) shows the deformation result with roundness parameter $\lambda = 5$ and only normal vectors as target directions ($a = 1$). To see the deformation result clear, Fig.4.11(c) is the face form of Fig.4.11(b). From Fig.4.11(b) and 4.11(c), we can observe that: (1) the eight corners of the square are overly growing, and the flat areas of the square are overly shrinking; (2) there are some concave

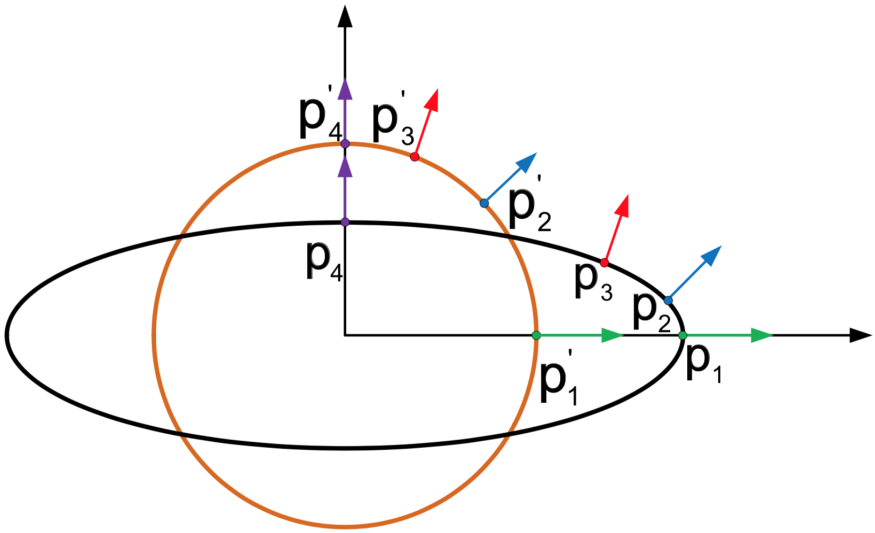
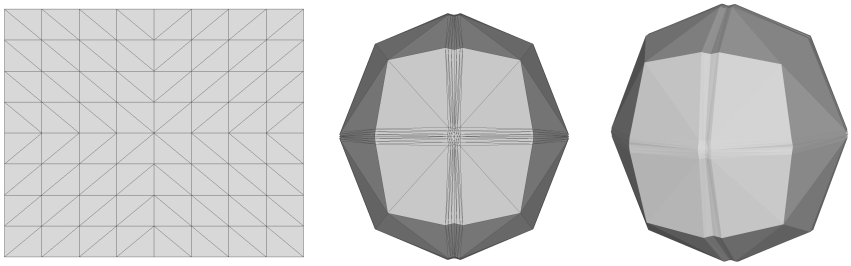


Figure 4.10: Analysis of overly growing and shrinking deformation

parts around the connection of the overly growing and shrinking areas.



(a) input (b) $\lambda = 5$ (c) face form

Figure 4.11: Deformation result of square model

Other models also have above two problems. Figure 4.12(a) is the input model. It is like a tetrahedron. Only normal vectors as target directions ($a = 1$), the deformed result is as Fig.4.12(b) with roundness parameter $\lambda = 0.5$. The deformation method deforms the convex sharp corners rounder. When λ is larger ($\lambda = 5$), the convex sharp corners are much rounder shown in Fig.4.12(c). At the same time, the convex sharp corners stretch largely and the flat parts become a little concave shown in Fig.4.12(b). With large

$\lambda(\lambda = 5)$, the deformation has more obviously over growing and shrinking areas shown in Fig.4.12(c). With large $\lambda(\lambda = 5)$, even mesh self-intersections exist due to overly growing shown in Fig.4.12(d).

Figure 4.13(b) is the deformation result with $\lambda = 0.1$. From Fig.4.13(b), the convex curved corners are deformed more obviously than other flat areas of the model. When given relatively large λ ($\lambda = 0.5$), the problem of overly growing even causes mesh self-intersections shown in Fig.4.13(c).

In summary, only normal vectors as target directions:

(1) the convex models, which consist of single component, the deformation method deforms the local convex curved surface rounder, such as Fig.4.7, 4.9, and 4.12(b).

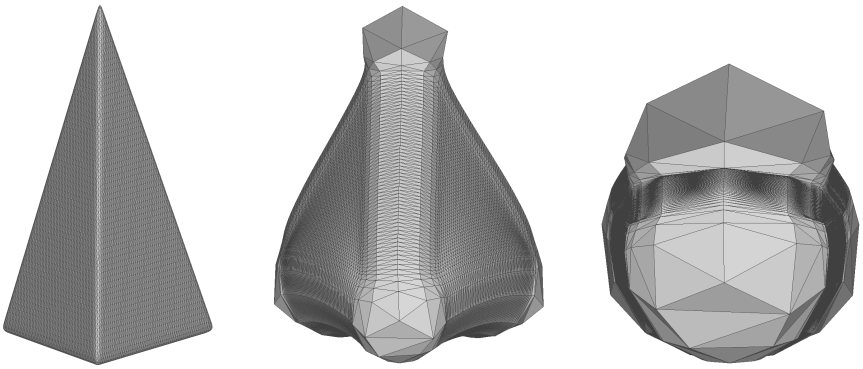
(2) with large λ, the deformation method causes overly growing and shrinking problem obviously. When λ is larger, overly growing problem causes mesh self-intersections.

4.5.3 Deformation of Complex Models

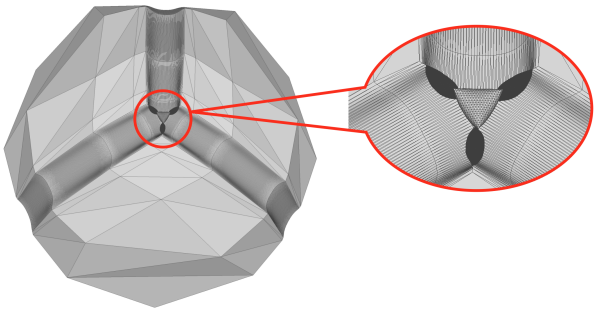
We experimented the deformation method on a complex model which contain concave and convex surfaces shown as Fig.4.14. Input is a bunny model as Fig.4.14(a). The deformation result with $\lambda = 0.1$ and $a = 1$ is as Fig.4.14(b). All components such as the ear, head, feet, and body, are rounder. However, mesh self-intersections occur from the neck to the back, where it appears as wrinkles. Therefore, for complex models, even with slightly concave surfaces, mesh self-intersections can occur with small roundness parameter λ.

4.5.3.1 Mesh Self-intersections

Figure 4.15 shows the discussion about mesh self-intersections of complex models. For concave surface (such as the neck part surface of the bunny

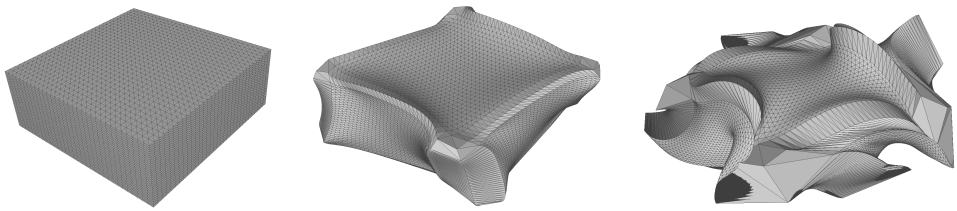


(a) input (b) $\lambda = 0.5$ (c) $\lambda = 5$



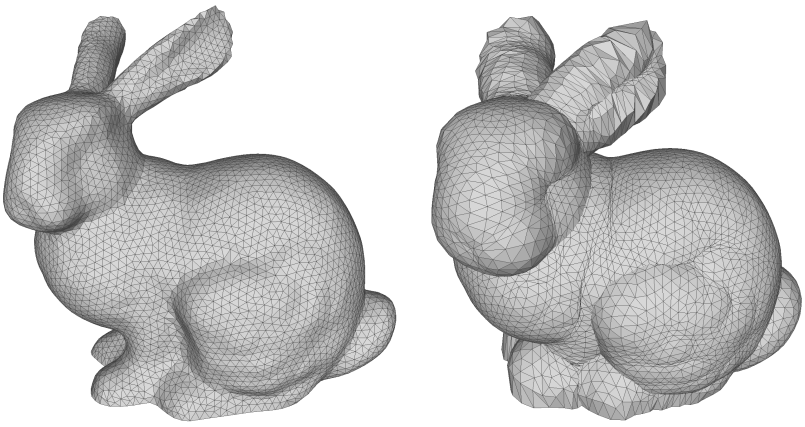
(d) $\lambda = 5$ (bottom view)

Figure 4.12: Deformation results of tetrahedron with normal vectors as target directions



(a) input (b) $\lambda = 0.1$ (c) $\lambda = 0.5$

Figure 4.13: Deformation results on Tangram with normal vectors as target directions



(a) input (b) $\lambda = 0.1$

Figure 4.14: Deformation of bunny model

model), represented by thick black line in Fig.4.15, there are two points $\mathbf{p}_1$ and $\mathbf{p}_2$. $\mathbf{n}_1$ and $\mathbf{n}_2$ are normal vectors of $\mathbf{p}_1$ and $\mathbf{p}_2$ respectively. $\Delta\mathbf{p}$ is the position vector difference represented by red line segment with arrow. $\Delta\mathbf{p} = \mathbf{p}_1 - \mathbf{p}_2$. $\Delta\mathbf{n}$ is the normal vector difference represented by red line segment with arrow. $\Delta\mathbf{n} = \mathbf{n}_1 - \mathbf{n}_2$. When $\Delta\mathbf{n}$ and $\Delta\mathbf{p}$ have opposite directions, minimization of $||r\mathbf{R}\Delta\mathbf{n} - \Delta\mathbf{p}||$ causes that the points $\mathbf{p}_1$ and $\mathbf{p}_2$ are very close and even let $\mathbf{p}_2$ locate at the left of point $\mathbf{p}_1$. When $\mathbf{p}_2$ is at the left of point $\mathbf{p}_1$, the mesh will generate self-intersections.

To avoid mesh self-intersections, when the angle is larger than 90° between normal vector difference $\Delta\mathbf{n}$ and the position vector difference $\Delta\mathbf{p}$, take $\Delta\mathbf{n}$ the opposite value as Eq.4.39. Also, the same operation in Eq.4.22 should be done when the angle is larger than 90° between normal vector difference $\Delta\mathbf{n}$ and the position vector difference $\Delta\mathbf{p}$. This method of avoiding mesh self-intersection is equivalent to changing the position of the center $\mathbf{O}$ of the sphere. The deformation of this method will make convex surfaces more convex, rounder and make concave surfaces more concave, rounder.

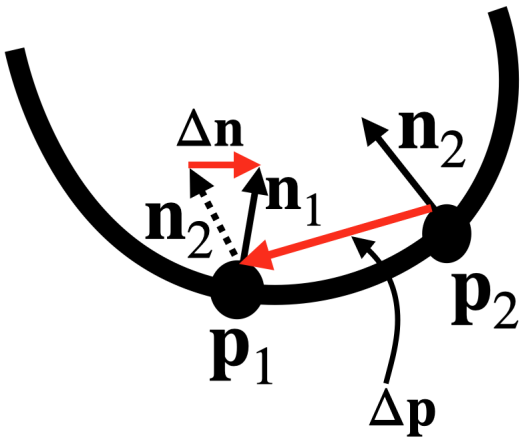
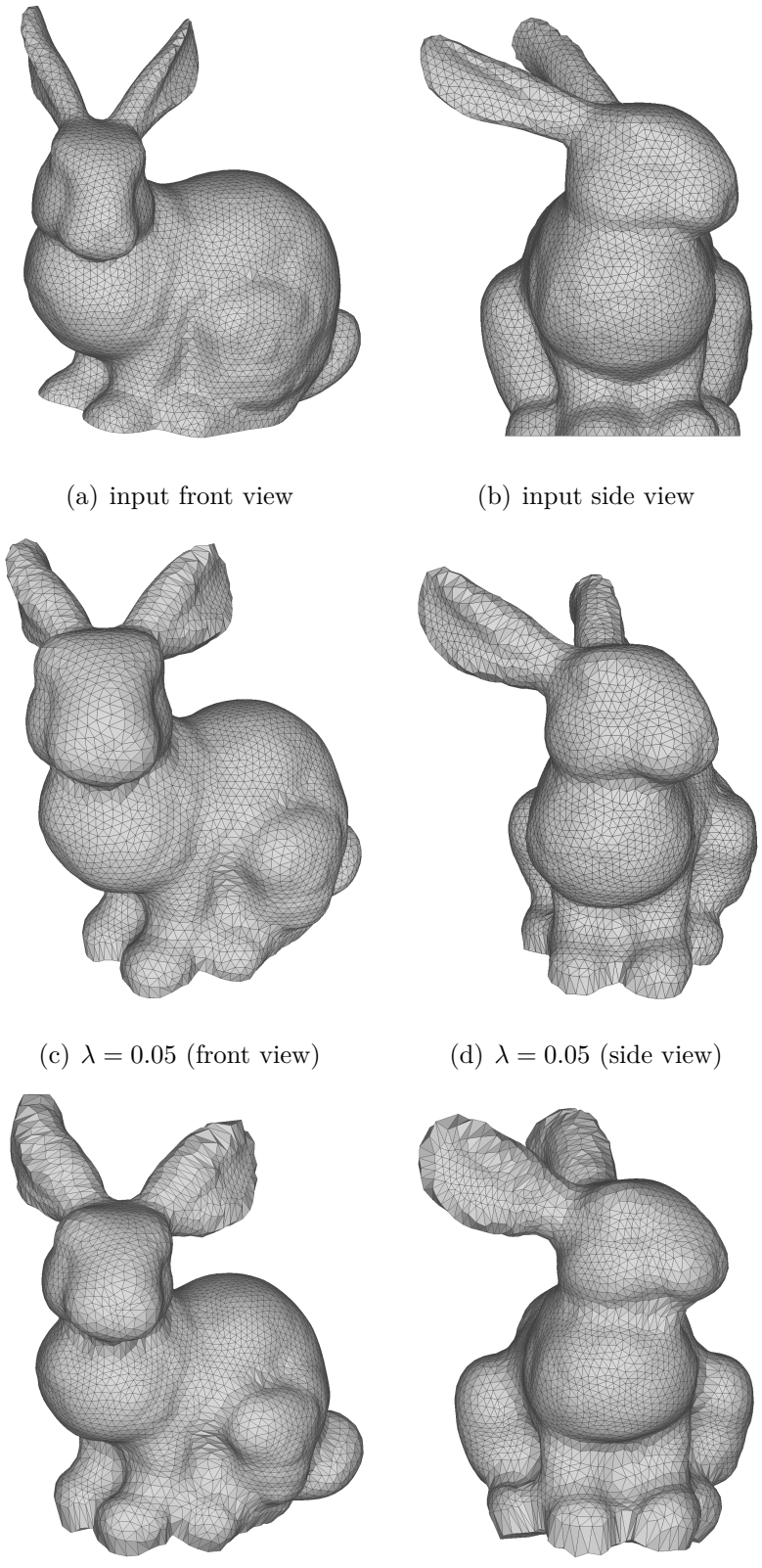


Figure 4.15: Discussion of mesh self-intersections for complex models

$$\begin{aligned} \Delta\mathbf{n} &= -\Delta\mathbf{n} \\ &= -(\mathbf{n}_1 - \mathbf{n}_2) \end{aligned} \tag{4.39}$$

The deformation result is shown in Fig.4.16. Input model from two views are shown in Fig.4.16(a) and 4.16(b). With $\lambda = 0.05$, the deformation result is shown in Fig.4.16(c) and 4.16(d). The concave surface parts are more concave and rounder, such as the areas around the neck and feet. The convex surface parts are more convex and rounder, such as the area below the head and the area around the feet. When $\lambda = 0.1$, the deformation effects are more obvious shown in Fig.4.16(e) and 4.16(f). Also the energy of this method still converges shown in Fig.4.17.

This method can avoid mesh self-intersections around concave surface for complex models. Currently, we have not found the applications of the deformation method of avoiding the mesh self-intersctions.



(a) input front view

(b) input side view

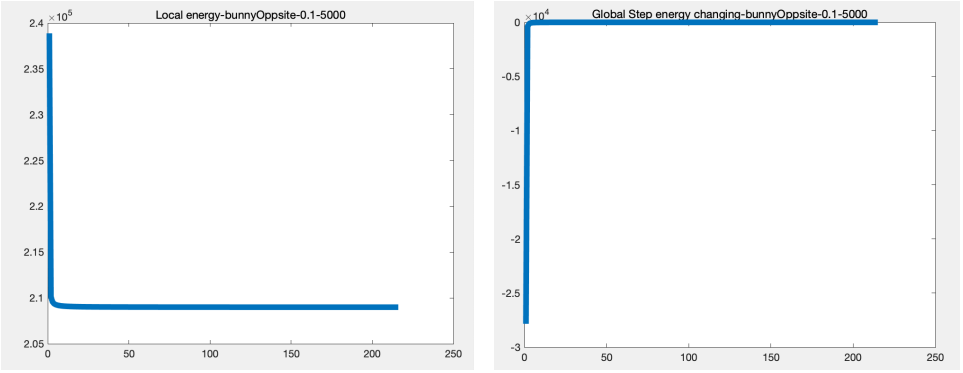
(c) $\lambda = 0.05$ (front view)

(d) $\lambda = 0.05$ (side view)

(e) $\lambda = 0.1$ (front view)

(f) $\lambda = 0.1$ (side view)

Figure 4.16: Deformation results of bunny (difference oppositely)



(a) local energy change (b) global energy change

Figure 4.17: Energy Changing of bunny (difference oppositely) with $\lambda = 0.1$

4.5.4 Discussion of Deformation Scale

Currently, the deformation scale of every point is λ. However, the local shape of every point is different. When adopt the normal vector as the spherical feature, it is not reasonable that the deformation scale of every point is λ. According to results, we can obtain the following experience. The deformation scale of point in curved surface has better be smaller than point in flat surface. The deformation scale of point in concave surface has better be smaller than point in convex surface.

Due to my limited ability, when normal vector is as the description of spherical feature, currently the exact deformation scale of every point is not sure. If deformation scale of every point is solved, maybe spherical style deformation of the entire model can be achieved.

4.6 Results with Position Direction

This section, we set only position directions as target directions, that is, $a = 0$. Simple and complex models are performed by this deformation method. The deformation results verify that this method with $a = 0$ has no mesh self-

intersections and can deform the models globally rounder. Then we discuss the reasons of no mesh self-intersections and show current limitations of this method.

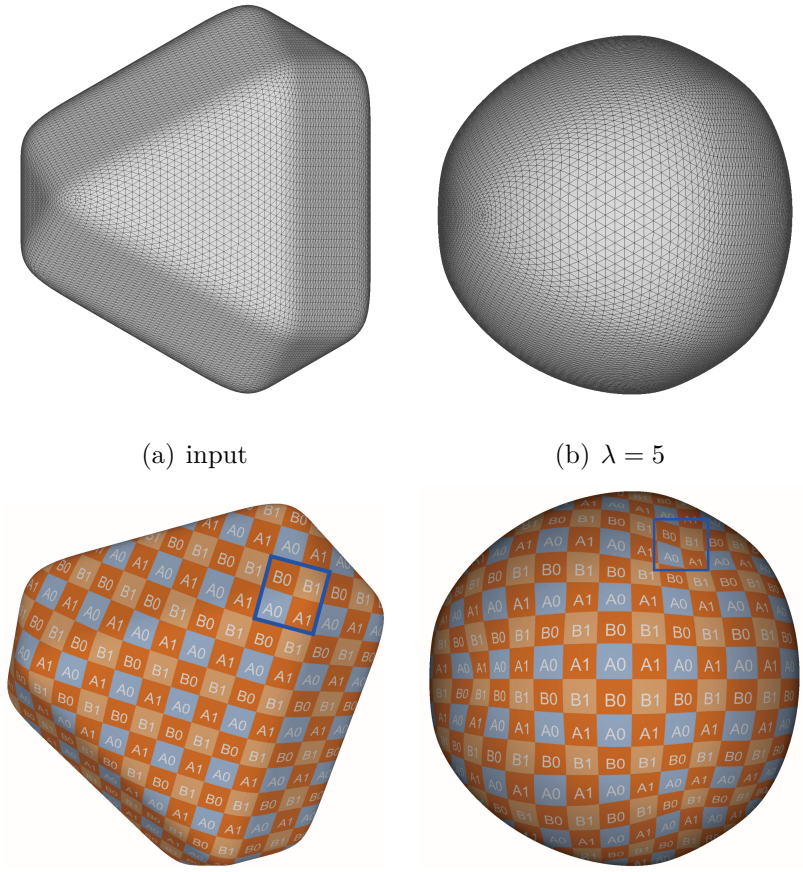
4.6.1 Deformation of Simple Models

We perform our deformation method on simple models, whose surfaces include no many details, with only position directions as target directions ($a = 0$).

Figure 4.18(b) shows the deformation result with $a = 0$ and $\lambda = 5$. Compared with Fig.4.9(b), Fig.4.18(b) shows the deformation result with more uniform mesh density. Obviously, it is no overly growing problem and mesh self-intersections. When a is equal to 0 (only position directions as target directions), the flat surfaces are rounder. The deformation method with only position directions as target directions ($a = 0$) deforms the whole model globally rounder.

Figure 4.19(b), 4.19(c), and 4.19(d) show different defroamtion results with same $a = 0$, but different λ values. Compared with Fig.4.12(c), and 4.12(d), Fig.4.19(d) shows the deformation result with more uniform mesh density and without overly growing probelm and mesh self-intersections. Again with the method of only position directions as target directions, the whole model becomes rounder, and when λ is larger, the deformation is rounder.

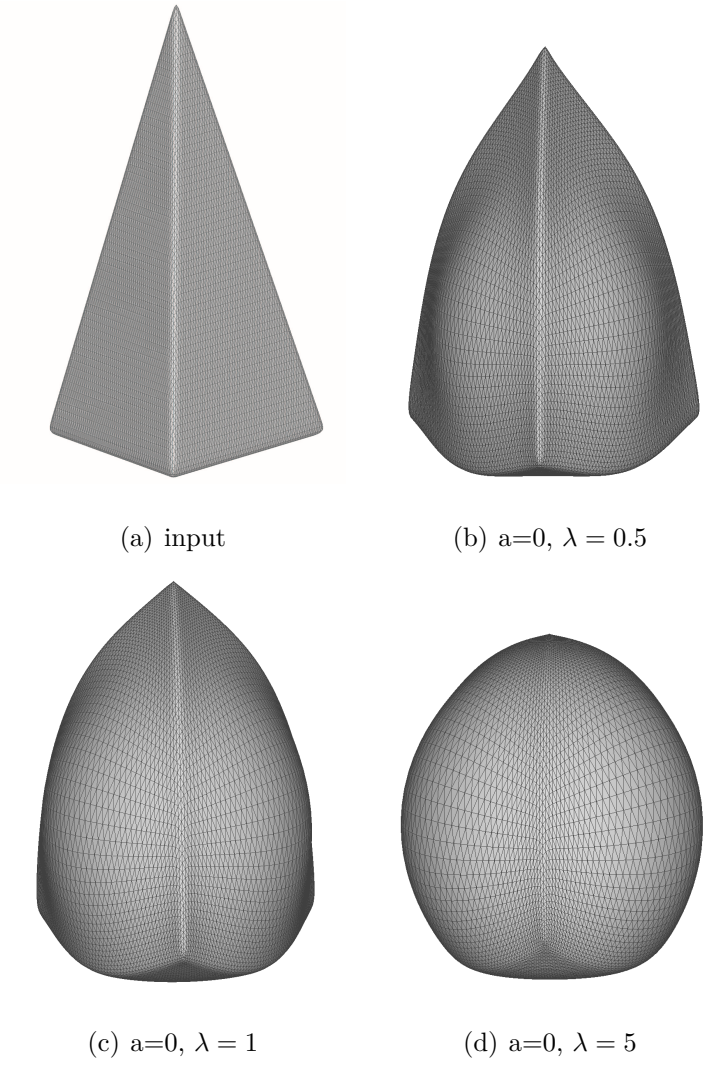
Figure 4.20 shows the deformation results of a cuboid shape, which has single component and few surface details. Figure 4.20(b), 4.20(c), and 4.20(d) are the deformed results with $\lambda = 0.1, 0.5, 5$ respectively. By setting $a = 0$, stable deformation has been conducted, no mesh distortion, and no mesh self-intersections, compared with Fig.4.13(b), and 4.13(c). As λ increases, the surface becomes rounder. In this case, we find that the vertical edges become slightly curved inward. While our algorithm generally works on regular



(a) input (b) $\lambda = 5$

(c) texture of (a) (d) texture of (b)

Figure 4.18: Deformation results of model4 (icosahedron) with $a = 0$



(a) input (b) a=0, $\lambda = 0.5$

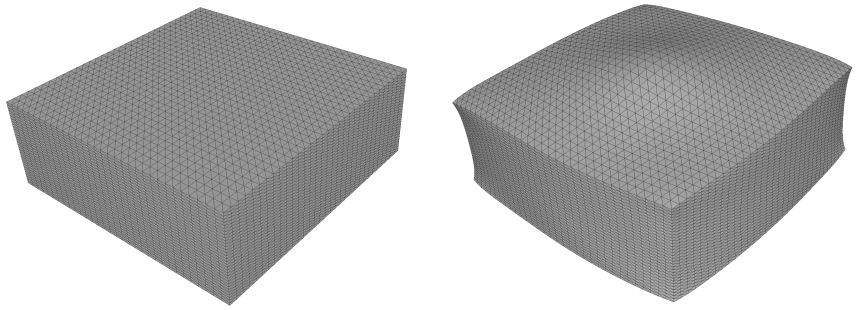
(c) a=0, $\lambda = 1$ (d) a=0, $\lambda = 5$

Figure 4.19: Tetrahedron deformation with $a = 0$

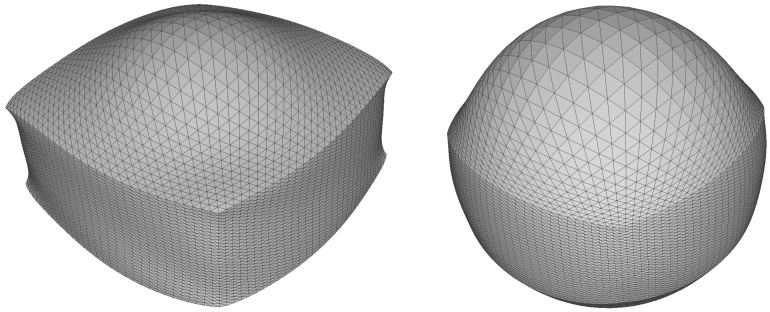
triangular meshes, the edge parts in the input model are composed of right triangles.

4.6.2 Deformation of Complex Models

We applied the spherical style deformation method with position directions as target directions on complex models which consist of many concave and convex surface details, and confirm that this method achieves deformed models without growing problem and mesh self-intersections. Complex models are



(a) input (b) $\lambda = 0.1$



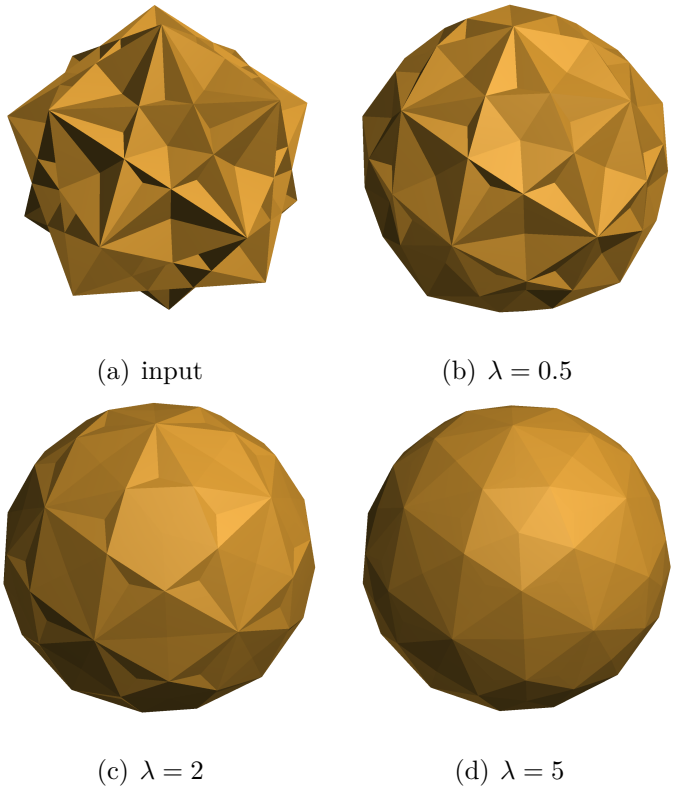
(c) $\lambda = 0.5$ (d) $\lambda = 5$

Figure 4.20: Cuboid deformation with $a = 0$ and different λ

deformed by this method in a simple way, adjusting parameter λ and sphere center $\mathbf{O}$. We show effects of λ and $\mathbf{O}$. Sphere center $\mathbf{O}$ is set as mean coordinate value of all points of a input mesh model by default. The models in this section are exported from Thingi10K.

We perform the deformation method on a model shown in Fig.4.21(a) (© Paul Moews under CC BY). As λ is larger, the surface is rounder. Figure 4.22(b) shows the deformation results of "female face" (© Anna Kaziunas France under CC BY). The deformed shapes overall look rounder, especially the face of the female. Meanwhile, the local details, such as the eyes, and mouth, are preserved. Sharp corners occurred in Fig.4.22(b), particularly in the upper right and lower right corners. This part of the input model contains triangles that are very thin and long.

We compare the deformation result of female face with the method of

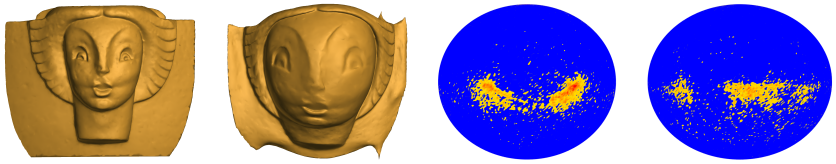


(a) input (b) $\lambda = 0.5$

(c) $\lambda = 2$ (d) $\lambda = 5$

Figure 4.21: Compound deformation with $a = 0$ and different λ

[14]. In the first row of Fig.4.23, these polyhedra are adopted as style models, because the surface normals of these polyhedra approximate the distribution of surface normals of the sphere. Deformation results are shown in the second row of Fig.4.23. In order to highlight the deformation results, we set style deformation weight parameter to a large value, such as 10. The face normals of style models are as target normal directions. As we discussed in section 2.2 Normal-based Approach, when the normal vector distribution of the style model is closer to the sphere's, the less the female face deforms. In Fig.4.23, style1-3 deform the female face into polyhedral styles, while style4 makes little deformation and is very similar to the input female face model. Our deformation result is noticeably better at making the face surface rounder than the results shown in Fig.4.23. We also used Gauss mapping to evaluate the deformation results. Gauss mapping reflects the normal vector distribution of



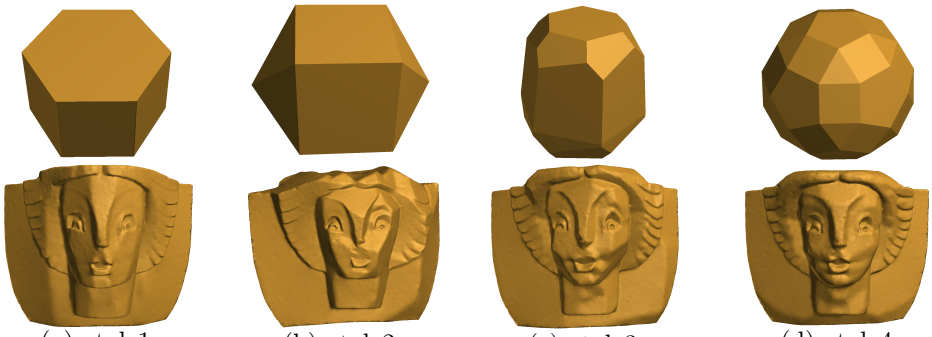
(a) input (b) $\lambda = 0.5$ (c) Gauss image of input (d) Gauss image of $\lambda = 0.5$

Figure 4.22: Female face deformation

a model. The blue color represents no distribution of normal vectors, while the yellow color represents a distribution of normal vectors. The redder the color is, the denser the distribution of normal vectors becomes. In Fig.4.24(a)-4.24(d), the distribution of normal vectors is equivalent to the distribution of yellow color and the red color represents the concentrated distribution of normal vectors. Compared with the results depicted in Fig.4.24, in Fig.4.22(d) the yellow color is more dispersed and red color is fewer. We also provided numerical comparison of Gauss mapping (shown in Table.4.1) to confirm above description. The sphere of Gauss mapping is divided into 360×180 units. The number of units in yellow of our result ($\lambda = 0.5$) is the most and the number of units in red of our result is the least. Our result exhibits a more dispersed and uniform distribution of normal vectors and closer to the normal vector distribution of the sphere. Therefore, our deformation result is better at making the female face surface rounder. In Fig.4.23, the style models are cited from © Chris under CC BY, © Paul Moews under CC BY, © roman jurt under CC BY, © Ann Eisenberg under CC BY from thingi10k respectively.

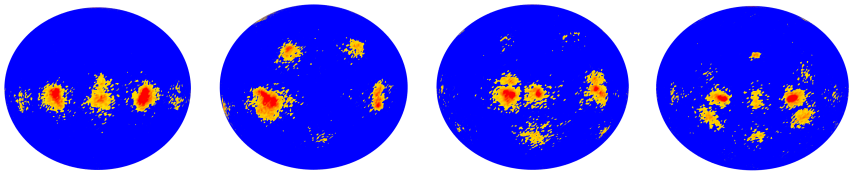
Figure 4.25(b) and 4.25(c) show the deformation results of "stone house" (© Perry Engel under CC BY). The deformed shape overall looks rounder, especially the roof of the stone house. Meanwhile, the local details, such as individual stone features, are preserved.

We also compare the deformation result of stone house with another rounder



(a) style1 (b) style2 (c) style3 (d) style4

Figure 4.23: Female face deformation by Normal-Driven Spherical Shape Analogies [14]



(a) style1 (b) style2 (c) style3 (d) style4

Figure 4.24: Gauss image of female face deformation by [14]

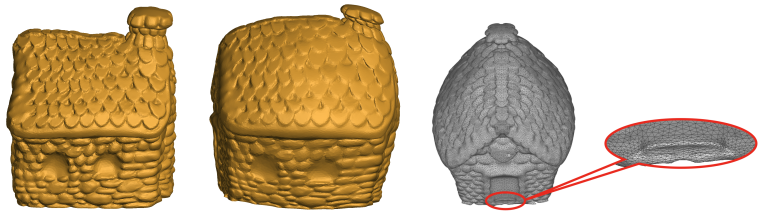
deformation method, outSphere method, shown in Fig.4.26. The outSphere method is as follows. Firstly, there is a sphere outside the house model shown in Fig.4.26(a). Start from the sphere center $\mathbf{O}$ to a point $\mathbf{p}_i$ on the stone house surface, and form a vector. This vector has an intersection point $\mathbf{q}_i$ with the sphere surface. Secondly, interpolate $\mathbf{p}_i$ and $\mathbf{q}_i$, then get the deformed result shown in Fig.4.26(b). Here interpolation coefficient is equal to 0.5. Compared with Fig.4.26(b), our result (shown in Fig.4.25(b)) is better to preserve the local shape of stones.

4.6.3 Sphere Center

The sphere center $\mathbf{O}$ can affect the deformation results. When $\mathbf{O}$ is closer to the surface, the surface is deformed rounder. In this subsection, we will show the effects of different sphere center $\mathbf{O}$.

Table 4.1: Numerical comparison of Gauss mapping

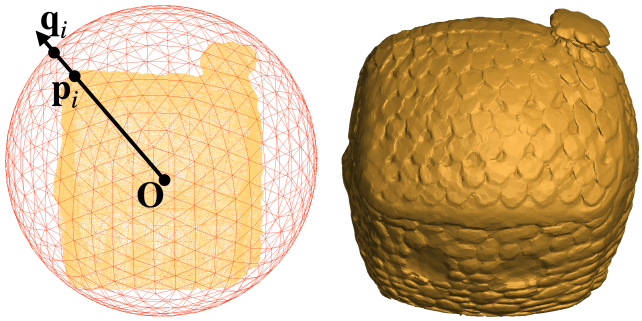
	the number of units in yellow	the number of units in red
input	2695	16
$\lambda = 0.5$	2938	0
style1	2145	113
style2	2398	91
style3	2527	54
style4	2438	56



(a) input (b) $\lambda = 0.5$ (c) mesh form

Figure 4.25: Stone house deformation

Figure 4.27 shows deformation results of changing the sphere center position with same fixed λ. We set the center position of the input shape (© Perry Engel under CC BY) to $y = 0$ in the vertical direction, and the deformation results, when y is changed up and down, are shown in Fig.4.27(b)-4.27(f). Here all center positions are inner of the doodlebot model. The closer the surface to the sphere center position, the rounder it is. In Fig.4.27(c), the center is closer to abdomen, and the abdomen surface becomes rounder. In Fig.4.27(f), the center is moved upper, closer to the top surface of the input doodlebot model, and the top surface is deformed much rounder. Figure 4.28 shows an example where the center position is set inside the mouth of the lion (© The Metropolitan Museum of Art). λ is larger, and the mouth is



(a) outSphere (b) deformation

Figure 4.26: Stone house deformation of out sphere method

deformed more.

4.6.4 Analysis of No Mesh Self-intersections

We explain reasons of distortion and mesh self-intersection from perspectives of local mapping and global mapping.

For smooth, convex models, the models do not have sharp corners or edges, and the distribution of normal vectors approximates the distribution of the normal vector of the sphere, such as the shape of a go stone. The deformation method with normal vectors as target directions has no mesh distortion and self-intersection problems for smooth, convex models.

However, for complex models, the distribution of normal vectors are complicated and different from the normal vector distribution of the sphere. Only normal vectors as target directions, the relationship is a local mapping, and lacks of the global mapping information. When the deformation optimization is performed, the energy of each point converges to a minimum, which corresponds to its local sphere mapping. When the parameter λ is larger, the local energy becomes larger, and it is easier to fall into a local minimum. Fig.4.29 illustrates our understanding. The black curve represents the surface, while the black vectors represent the normal vectors on the surface. $\mathbf{p}_i$ and $\mathbf{p}_k$ are

two points on the surface, and their normal vectors are represented by red and blue vectors, respectively. Only the normal vector mapped as the target normal direction projects $\mathbf{p}_i$ and $\mathbf{p}_k$ onto their respective spheres. Therefore, for deformations of complex models, it is easy to exist mesh distortions and self-intersections. When λ is larger, the distortions and self-intersections are more.

The position direction is related with the sphere center and the position of each point. The mapping of position directions as target directions is global mapping relationship. The global mapping redefines the distribution of normal vectors of all points. Therefore, the deformation with position direction has no mesh self-intersections, no distortion. Using interpolation methods (normal vector and position direction interpolation) can alleviate mesh self-intersections and distortions shown in next section.

From the view of energy optimization, in Fig.4.10, when roundness parameter λ is large, there are possibilities that the growing level of the area between $\mathbf{p}'_1$ and $\mathbf{p}'_2$ is faster than the shrinking level of the area between $\mathbf{p}'_3$ and $\mathbf{p}'_4$. Therefore, there are possibilities that after deformation, $\mathbf{p}'_2$ will be located between $\mathbf{p}'_3$ and $\mathbf{p}'_4$. When $\mathbf{p}'_2$ is located between $\mathbf{p}'_3$ and $\mathbf{p}'_4$, it causes mesh self-intersetions such as mesh self-intersection shown in Fig.4.12(d) and 4.13(c). Similarly, in Fig.4.15, when λ is larger, after deformation optimization, there are possibilities that $\mathbf{p}_2$ will be located at the left of $\mathbf{p}_1$. When $\mathbf{p}_2$ is lacated at the left of $\mathbf{p}_1$, it causes mesh self-intersections such as Fig.4.14(b).

When minimizing $||r\mathbf{R}_i(\mathbf{n}_i - \mathbf{n}_j) - (\mathbf{p}_i - \mathbf{p}_j)||^2$, in Fig.4.15 it is easy to cause mesh self-intersections. Position directions as target directions can avoid mesh self-intersections which happened in Fig.4.15. Figure 4.30 shows position direction difference and position vector difference. $\mathbf{O}$ is the sphere center. $\mathbf{d}_1$ and $\mathbf{d}_2$ are position directions of point $\mathbf{p}_1$ and $\mathbf{p}_2$ respectively, which are unit

vectors from $\mathbf{O}$ to $\mathbf{p}_1$ and $\mathbf{p}_2$. $\Delta\mathbf{d}$ is the difference of position directions and $\Delta\mathbf{d} = \mathbf{d}_1 - \mathbf{d}_2$. $\Delta\mathbf{p}$ is the defference of position vectors and $\Delta\mathbf{p} = \mathbf{p}_1 - \mathbf{p}_2$. When minimizing $||r\mathbf{R}_i(\mathbf{d}_i - \mathbf{d}_j) - (\mathbf{p}_i - \mathbf{p}_j)||^2$, the directions of $\Delta\mathbf{d}$ and $\Delta\mathbf{p}$ are not opposite. Therefore, when optimizing energy with position directions as target directions ($a = 0$), the deformation are relatively uniform mesh density compared with the method with normal vectors as target directions ($a = 1$), and there are no mesh self-intersections.

4.6.5 Global Mapping of One Sphere Center

In section 4.6.3 and 4.6.2, we find the stone house chimney shorter, the doodlebot feet flatter, and the lion nose flatter, after deformation. We also did experiments with one sphere center on bunny model, which consists of more than one component model. Because As λ is larger, the deformation is closer to a sphere, shown in Fig.4.32.

Figure 4.33 shows the reasons. Black line represents input bunny model. $\mathbf{O}$ is the center point of bunny model. Connect $\mathbf{O}$ and each point on surface, and obtain unit target directions shown as blue lines with arrow. There are three points $\mathbf{A}, \mathbf{B}, \mathbf{C}$ on the same blue line. $\mathbf{A}, \mathbf{B}, \mathbf{C}$ have same unit target direction. After optimization, the volume between $\mathbf{B}$ and $\mathbf{C}$ will become thinner. If the controlling parameter λ is very large, such as $\lambda = 10$, $\mathbf{B}$ and $\mathbf{C}$ will approximately touch onto one same plane. Similarly, after optimization, the distance between $\mathbf{A}$ and $\mathbf{B}$ will be shorter, and the volume $\mathbf{OA}$ will be bigger.

4.6.6 Limitation of Multi-component Model

Currently, we focus on spherical style deformation of a single-component model. Our method can also be applied to multi-component models, converg-

ing to multiple spheres. Fig.4.34(a) shows the input model (3D EXPORT, © nikakihnocapov under CC BY). Firstly, the head component is selected manually and deformed. Then, the sphere center is determined based on the average coordinate of the points belonging to the head component. Similarly, the body and feet components are deformed. Fig.4.34(b) shows the deformation result. Our spherical style deformation algorithm is suitable for multi-component models after setting multiple sphere centers and deforming them step-by-step.

Currently, components are selected manually. In the future, we will develop a convenient and real-time interactive spherical style deformation tool for multi-component models.

4.7 Results with Interpolation Direction

When a is equal to 1, the method deforms the local curved surface rounder. When a is equal to 0, the method deforms the flat surface rounder and can deform the whole model globally rounder. When $0 < a < 1$, target directions are linear combination of normal vectors and position directions. The deformation results are in between $a = 0$ and $a = 1$. When $0 < a < 1$, there is no guarantee that the deformed results will not have mesh self-intersections.

Figure 4.35(b) is the deformation result with $a = 0.5$ and $\lambda = 5$. Compared with Fig.4.9(d) and 4.18(d), the deformation result (Fig.4.35(d)) is in between.

Compared with Fig.4.12(b) and 4.19(b), Fig.4.36(b) is also in between. In Fig.4.37, the larger the value of a, the rounder the parts such as ears, face and legs are deformed.

4.8 Discussion

In this section, we will discuss the convergence and stability of matrix $\mathbf{L}_0$ (as Eq.4.27).

4.8.1 Convergence

The convergence of energy function Eq.4.8 will be discussed. The energy function Eq.4.8 is obtained by summing the energy of each point. Therefore, the convergence of energy function of a point, such as the ith point shown as Eq.4.9, is analyzed firstly.

For cell C_i, in every local step optimization, given $\mathbf{S}_i$ (shown as Eq.4.13), it is a linear optimization to solve the rotation matrix $\mathbf{R}_i$ [19].

Figure 4.38 shows the optimization process of cell C_i. Initially, $\mathbf{S}_{i0}$ is given. With linear optimization (local step), obtain the closest rotation $\mathbf{R}_{i1}$. Given $\mathbf{R}_{i1}$, updating points' positions $\{\mathbf{p}_i\}$ is also linear optimization (global step). After update positions and get new $\mathbf{S}_i$, that is, $\mathbf{S}_{i1}$. Then repeat above iterations (local step and global step). Note that after update positions, $\mathbf{S}_{i1}$ and $\mathbf{S}_{i0}$ are most likely different.

After every local step, the energy of each cell C_i will not increase. Before and after every global step, the energy of the entire model will not increase. However, before and after every global step, there is no guarantee that the energy of each cell C_i will not increase. The energy of some cells may decrease faster. The energy of some cells may decrease slower. In the global step of every iteration, the mesh is a whole, the connection relationship of points is complicated and points are related and influence. In general, in every global iteration, it is not recommended to analyze each cell of the mesh as an individual. In a word, in every optimization step, such as local step and global step, the energy of the entire model is not increased. So the energy converges.

In our experiments, it is not evident that energy convergence fails, same as [20]. The whole optimization, consisting of local step optimzation and global step optimization, is to obtain the minimum energy value of the entire model.

However, the optimized $\mathbf{R}_i$ is related with the initial value $\mathbf{e}'_{ij}$, λ, $\mathbf{O}$. When the initial values are different, the final minimum value of the energy for C_i (shown in Eq.4.9) might be different. Though every optimization step is linear optimization in the whole optimization process, the energy is non-linear, multiple local minima may exist and the solution depends on the initial guess, same as [20]. In this research, initial value $\mathbf{e}'_{ij} = \mathbf{e}_{ij}$, which is similar to [13].

4.8.2 Stability

Condition number is used to measure matrix stability. Usually, Eq.4.40 is used to evaluate the conditioning of matrix $\mathbf{A}$. $\sigma_{\max}$ is the maximum singular value of $\mathbf{A}$. $\sigma_{\min}$ is the minimum singular value of $\mathbf{A}$. There is one caveat that prevents Eq.4.40 for the condition number from being used universally. In some cases, algorithms for computing $\sigma_{\min}$ may involve solving systems $\mathbf{Ax} = \mathbf{b}$, a process which in itself may suffer from poor conditioning of $\mathbf{A}$. Hence, we can not always trust values of $\sigma_{\min}$ [19].

$$cond\ \mathbf{A} = \frac{\sigma_{\max}}{\sigma_{\min}} \tag{4.40}$$

As far as we know, in [20], the condition number of Laplacian matrix is generally proportional to the mesh size. If refine the mesh, the condition number will grow proportionally. This means as the meshes are refined stability deteriorates [20].

In our research, we can not confirm that Eq.4.40 is completely proper for the condition number calculation. Meanwhile, the number of iterations

required to get reasonably close to a minimum depends on the condition number of the matrix $\mathbf{L}_0$ [20].

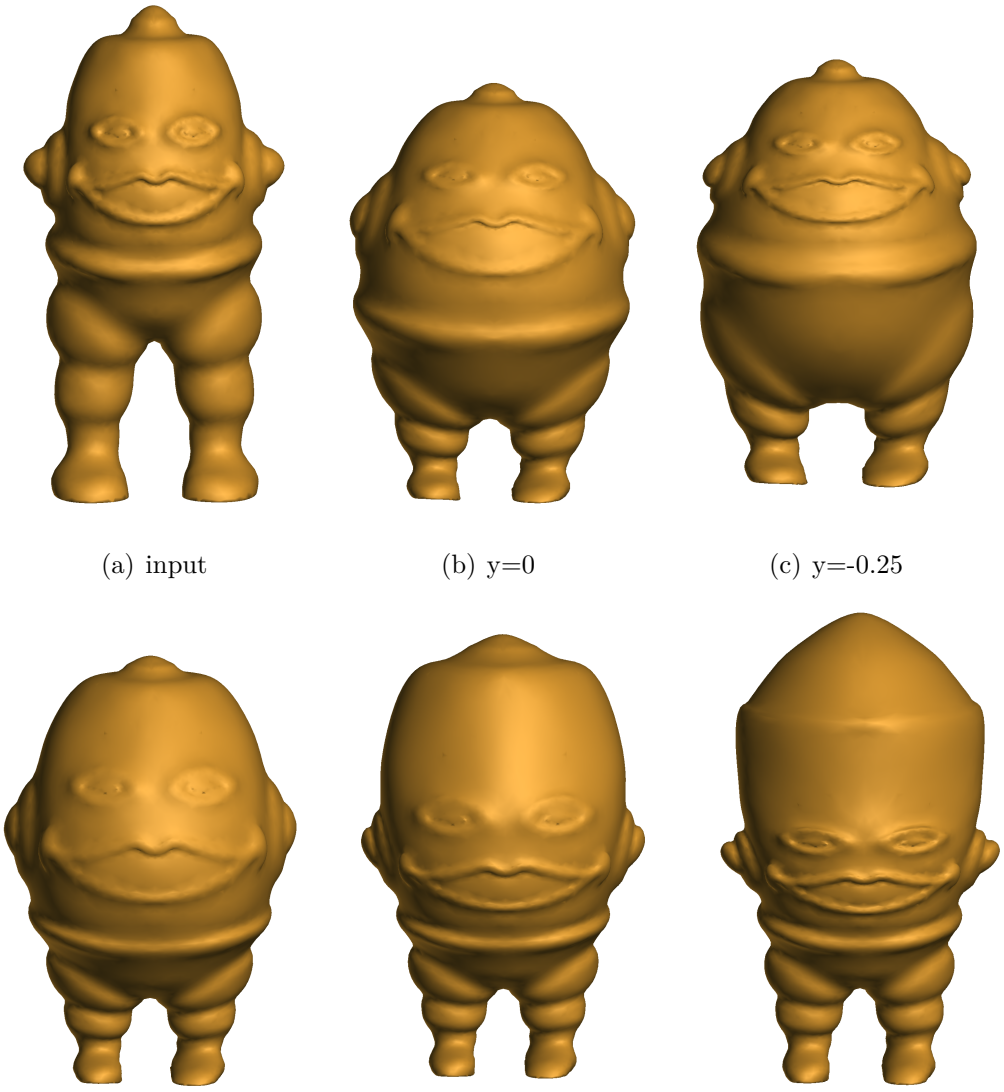
In our experimental results, when the mesh has more points, the iteration number is more. In Fig.4.39, Fig.4.39(a) and 4.39(b) are deformation results with same input shapes and deformation parameters, except for mesh size. In Fig.4.39(c) and 4.39(d), the horizontal axis, that is, the x-axis represents the number of iterations, and the vertical axis represents the energy.

Therefore, mesh size affects the condition number of matrix $\mathbf{L}_0$, and as the meshes are refined stability deteriorates. In this reasearch, we experimented on most not big size meshes. In future, we will continue to explore the stability of the matrix $\mathbf{L}_0$.

4.9 Summary

The deformation method of position directions as target directions deforms the flat surfaces rounder and the deformed whole model overall looks rounder. Oppositely, the deformation method of normal vectors as target directions deforms local curved surfaces rounder. Interpolation of normal vectors and position directions as target directions is another redefinition of the distribution of normal vectors.

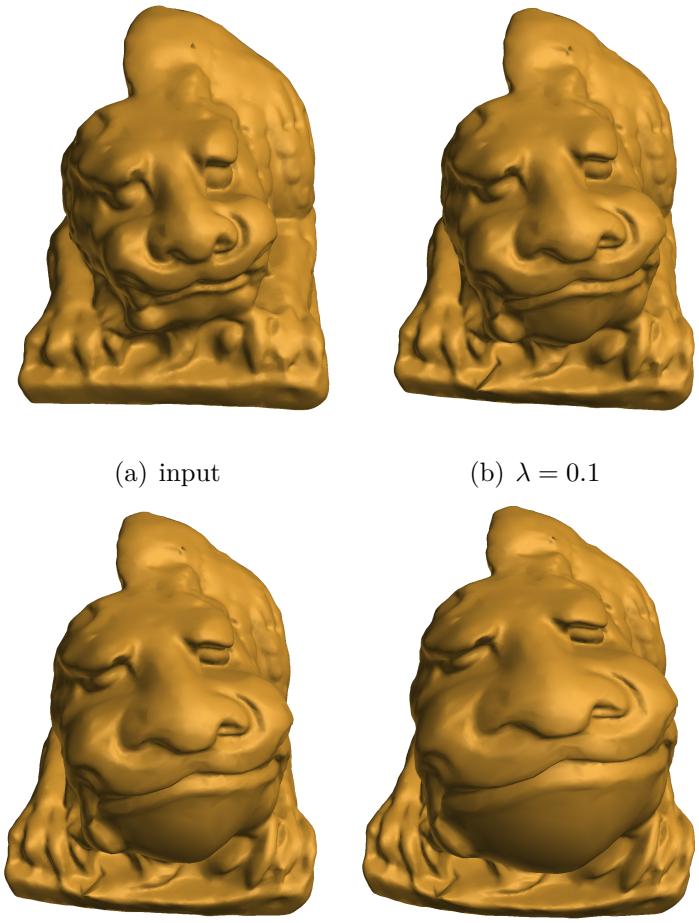
In future, there are possibilities for an application of local surface editing. The deformation method, whose target directions are set to normal vectors, or the interpolation of normal vectors and position directions, has the deformation effects of a simple brush tool. Users could use the tool to edit local surfaces, and achieve variable editing effects by adjusting the parameter a.



(a) input (b) y=0 (c) y=-0.25

(d) y=0.25 (e) y=0.5 (f) y=0.75

Figure 4.27: Doodlebot deformation with different sphere centers



(a) input (b) $\lambda = 0.1$

(c) $\lambda = 0.2$ (d) $\lambda = 0.3$

Figure 4.28: Lion deformation with different λ

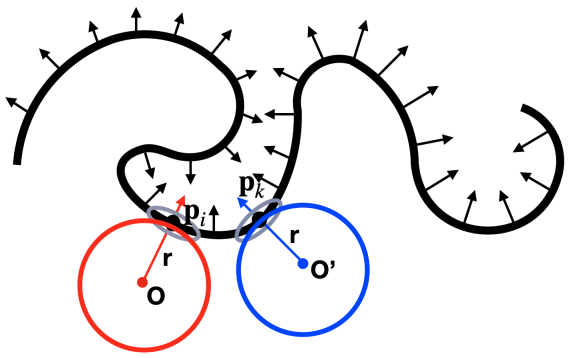


Figure 4.29: Local mapping

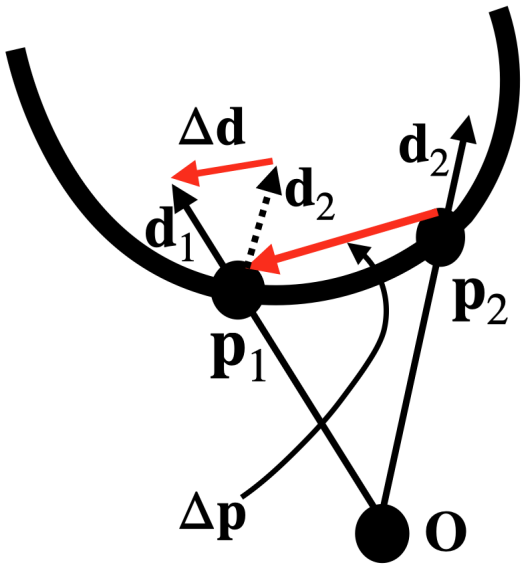
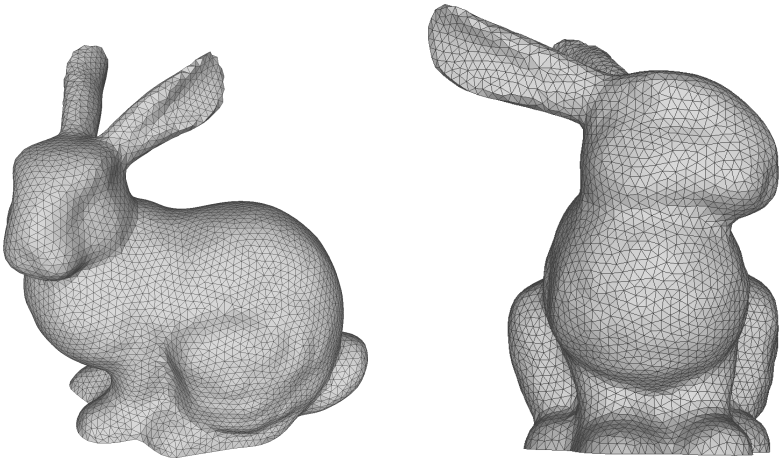
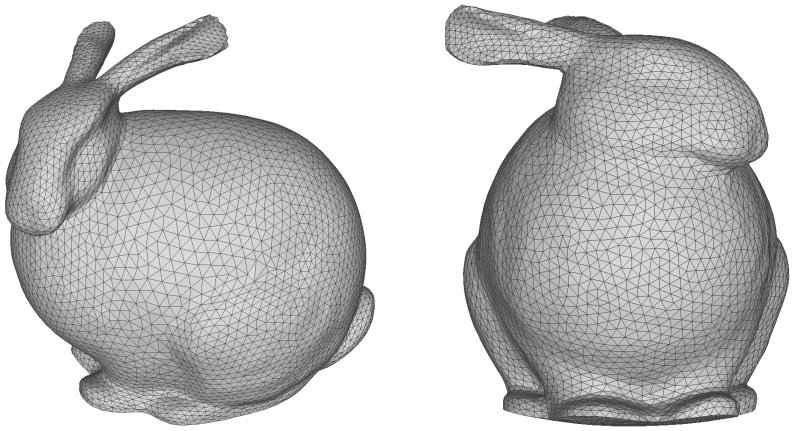


Figure 4.30: Difference between position directions and position vectors



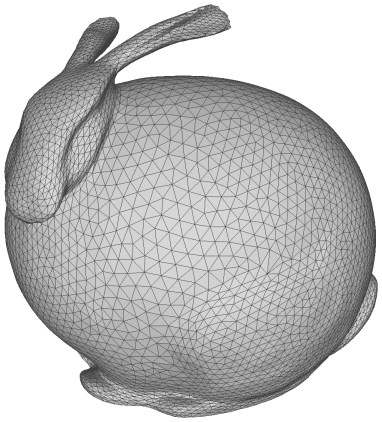
(a) input front view (b) input side view

Figure 4.31: Input model of bunny

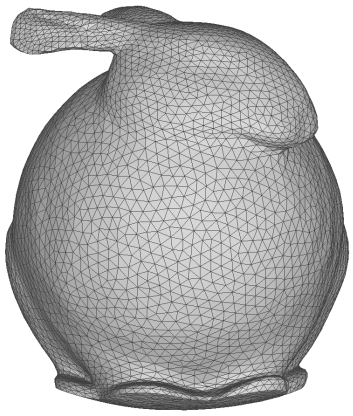


(a) $\lambda = 0.5$ (front view)

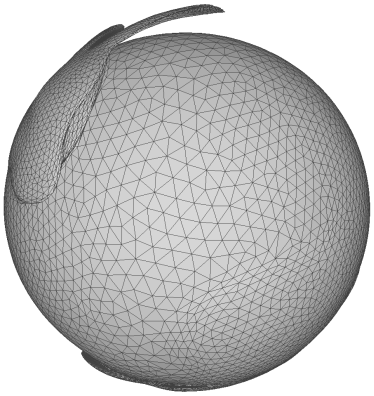
(b) $\lambda = 0.5$ (side view)



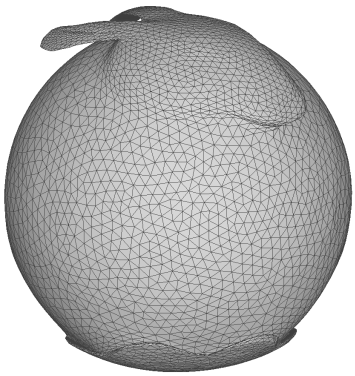
(c) $\lambda = 1$ (front view)



(d) $\lambda = 1$ (side view)



(e) $\lambda = 5$ (front view)



(f) $\lambda = 5$ (side view)

Figure 4.32: Deformation results of bunny with one sphere center

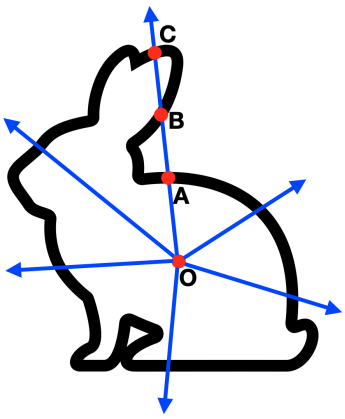
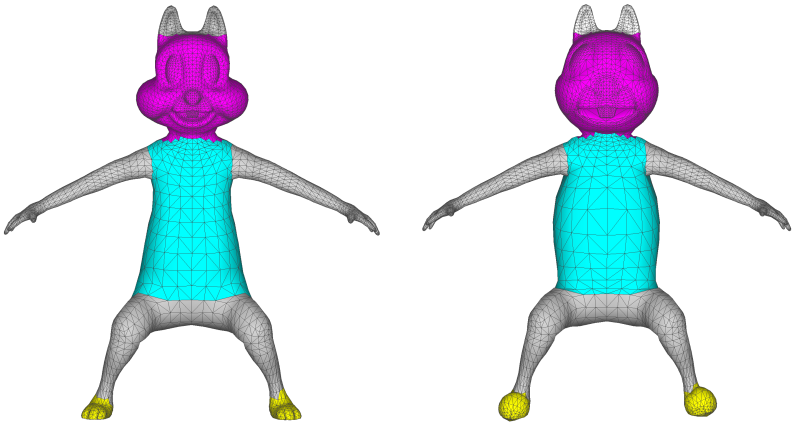
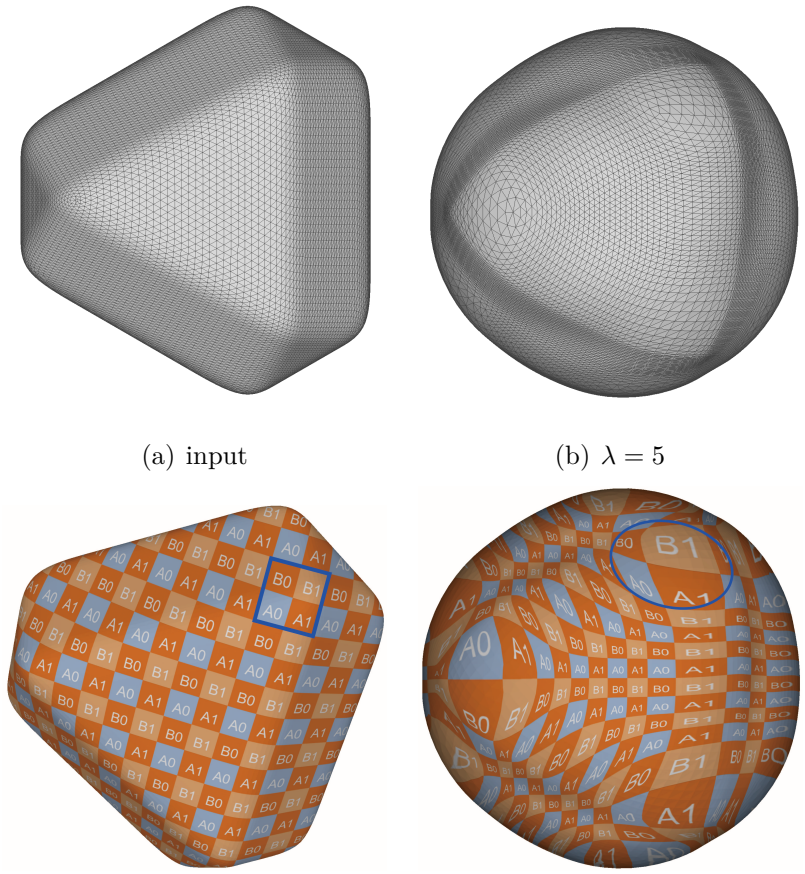


Figure 4.33: Deformation result analysis of thinner parts



(a) input (b) result

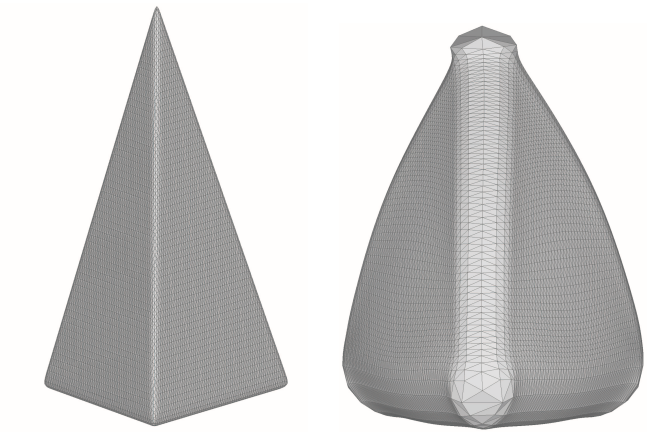
Figure 4.34: Multi-component model deformation



(a) input (b) $\lambda = 5$

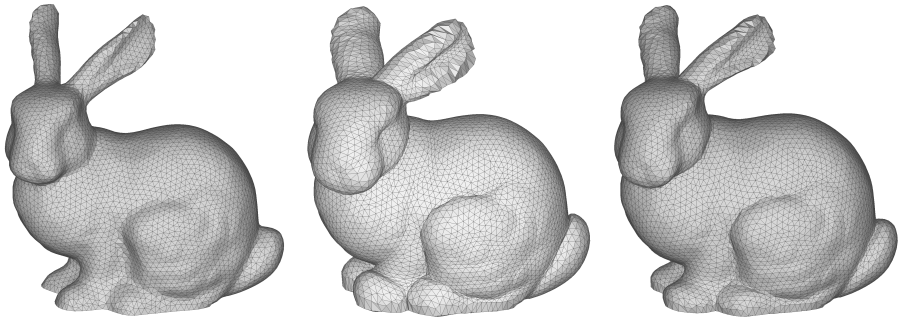
(c) texture of (a) (d) texture of (b)

Figure 4.35: Deformation results of model4 (icosahedron) with $a = 0.5$



(a) input (b) a=0.5, $\lambda = 0.5$

Figure 4.36: Tetrahedron deformation with $a = 0.5$



(a) input (b) $a = 0.5$, $\lambda = 0.1$ (c) $a = 0.3$, $\lambda = 0.1$

Figure 4.37: Deformation of bunny with $a = 0.3, 0.5$

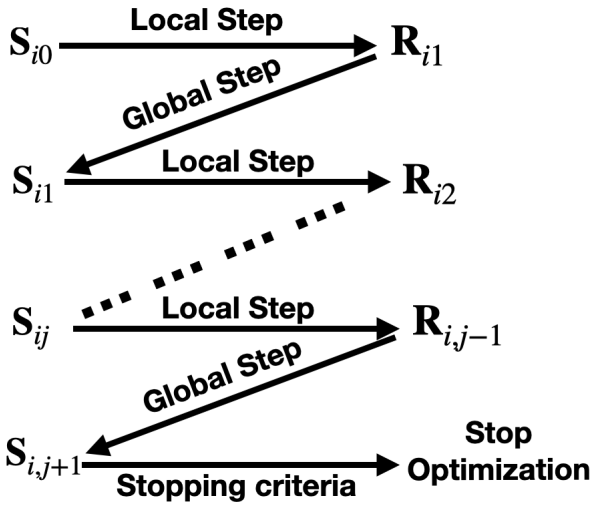
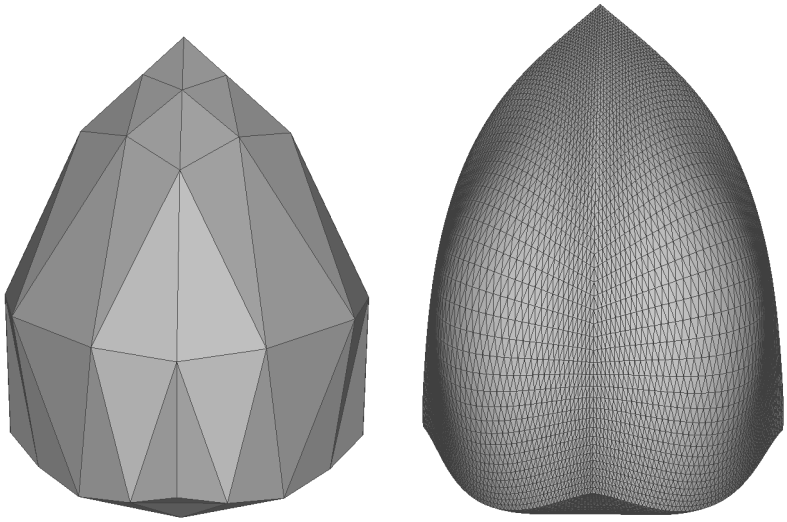
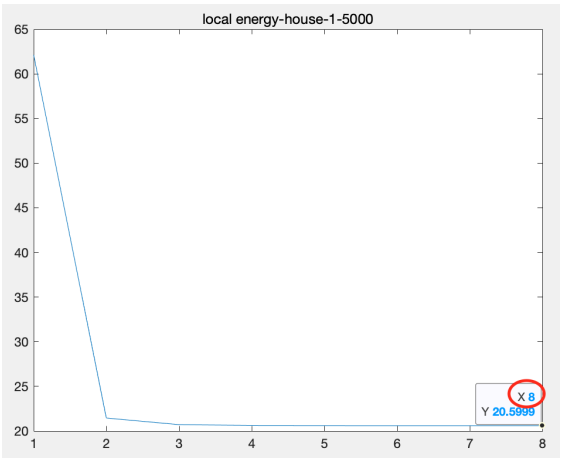


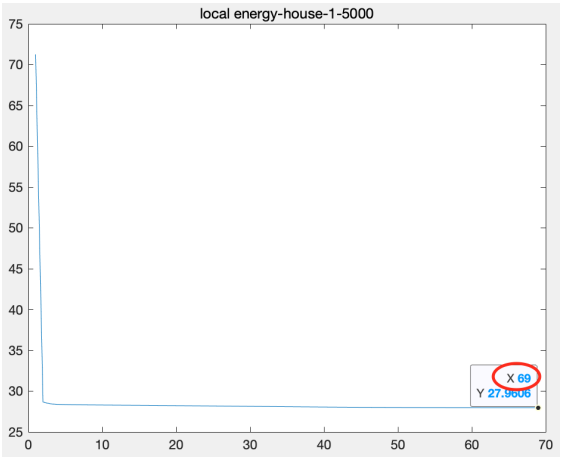
Figure 4.38: The whole optimization process of cell C_i



(a) few points (b) more points



(c) iteration number of (a)



(d) iteration number of (b)

Figure 4.39: Iteration numbers of different mesh size

Chapter 5

Proxy Model Method

5.1 Overview

Chapter 4 described the spherical style deformation method directly applied to the input models. However, the deformation results are dependent on the quality of the input mesh. The matrix ($\mathbf{L}_0$) still has symmetric, sparse, and positive semi-definite properties for regular triangular meshes.

A discrete Laplacian is a "perfect" discrete Laplacian that satisfies (SYMMETRY)+(LOCALITY)+(LINEAR PRECISION)+(POSITIVE WEIGHTS) if and only if the triangulation is regular [22]. When the input mesh is obtuse triangles, $\mathbf{L}$ does not satisfy the positive weights property, resulting in a non-positive definite matrix. This violates the discrete maximum principle [22], leading to potential oscillation of the numerical method [1]. With obtuse triangular mesh, $(w_{ij} + \lambda)$ in Eq. (4.22) can not be guaranteed to be all positive, even with the term of λ. Obtuse triangles lead to poorly conditioned matrices, worsening the speed and accuracy of the linear solver [1]. When the input mesh is obtuse triangular mesh, the spherical style deformation method often fails to achieve satisfactory results.

To solve this problem, one candidate technique might be locally remesh-

ing. The main steps are as follows. Inside the obtuse triangle, insert new points. Remesh new regular triangles using the inserted and original points of the obtuse triangle. But after remeshing, t-junction problem might appear. However, when t-junction exists in our deformation process, it is difficult to calculate the cotangent weight. The requirement without t-junction might lead locally remeshing complicated.

Currently we focus on model deformation, so we propose the deformation method based on convex hull proxy model as the complementary deformation method. Our method firstly constructs the proxy model of the input model and applies our deformation method to the proxy model. Then, the input model is deformed by projection and interpolation. Steps of the method are shown in following sections.

5.2 Proxy Model

The proxy model is constructed as follows:

(1) The convex hull of the input model is constructed firstly.

(2) Then, Poisson-disk sampling algorithm [5] is applied to the convex hull.

(3) Ball-Pivoting algorithm [2] is performed on these sampling points, and a proxy model, consisting of most regular triangles, is constructed. The proxy model is not required to be fully convex.

Figure 5.1 shows the overview of proxy model method. The input model is a car and its proxy model is represented by the blue line. Convex hull is the simple model that is closer to the input model. Therefore we use the approximately convex hull as the proxy model.

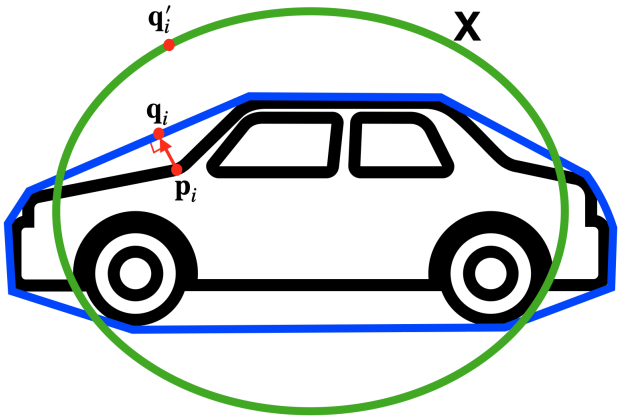
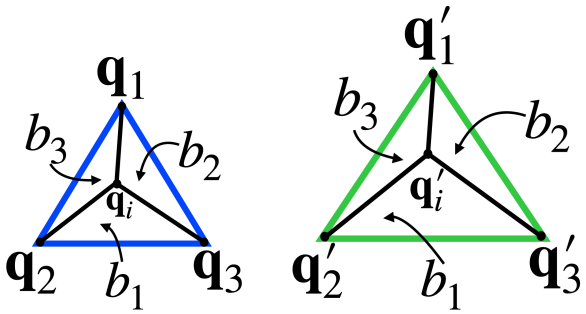


Figure 5.1: Overview of proxy model method

5.3 Projection Calculation

project the points of the input model onto the proxy model. Then obtain the corresponding projection points. Calculate the interpolation coefficients b_1, b_2, b_3 in barycentric system for each projected point within the triangle on the proxy model. In Fig.5.1, $\mathbf{p}_i$ is the ith point of the input model. $\mathbf{q}_i$ is the projection point of $\mathbf{p}_i$ on the proxy model. $\mathbf{q}_i$ is within the triangle $\triangle\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ on the proxy model shown in Fig.5.2(a), and satisfies the formula $\mathbf{q}_i = b_1 * \mathbf{q}_1 + b_2 * \mathbf{q}_2 + b_3 * \mathbf{q}_3$.



(a) $\triangle\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ (b) $\triangle\mathbf{q}_1'\mathbf{q}_2'\mathbf{q}_3'$

Figure 5.2: Interpolation coefficients b_1, b_2, b_3

5.3.1 Projection

We calculate the projection that the point of input model projects to the proxy model. the position of the ith point of the input model is as $\mathbf{p}_i$.

(1) Firstly, calculate k nearest neighbors of point $\mathbf{p}_i$ on the proxy model and record the k nearest neighbors as the set K_i.

(2) Secondly, for every point $CP_j \in K_i$, find its triangle faces CF_i on the proxy model where point CP_j belongs to. The triangle faces of all points in K_i constitute the candidate projection triangle faces recorded as the set SF_i.

(3) Thirdly, for every triangle face $f_l \in SF_i$, calculate the projection point of $\mathbf{p}_i$ on triangle face f_l. If the projection point is in the triangle face f_l or on the edge of the triangle face f_l, the projection point is the right projection point. Otherwise, it is not the right projection point.

(4) Fourthly, according to (3), there are three cases:

(I) First case: finally, we get one projection point. Then the projection point is the final projection point of $\mathbf{p}_i$ and denote the final projection point as q_i. At the same time, the face index, and linear interpolation coefficients of the projection point q_i on the triangle face are saved.

(II) The second case: finally, we get more than one projection points. Then select the projection point with the shortest projection distance as the final projection point q_i. Also the face index, and linear interpolation coefficients of the projection point q_i on the triangle face are saved.

(III) The last case: finally, we don't get any projection point in or on the triangle face. Then we select the alternative projection point with the shortest projection distance from not right projection points according to (3). And calculate the projection point of the alternative projection point by the same projection method according to (2),(3),(4). But for simple calculation, we just set nearest neighbors $k = 4$.

We can get corresponding projection point for each point of the input model. Next section we show the calculation process of linear interpolation coefficients.

5.3.2 Interpolation coefficients Calculation

We suppose the projection point as $\mathbf{q}_i$, and $\mathbf{q}_i$ is in triangle $\Delta\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$. In this section, to convenient writing, $\mathbf{q}_i$ is replaced with $\mathbf{p}$. b_1, b_2, b_3 are replaced with i, j, k respectively. $\Delta\mathbf{q}_1\mathbf{q}_2\mathbf{q}_3$ is replaced with $\Delta\mathbf{ABC}$.

Next, we will show the calculation process of interpolation coefficients i, j, k [10]. We suppose the projection point $\mathbf{p}$ is in triangle $\Delta\mathbf{ABC}$. Then we have Eq.5.1. i, j, k are interpolation coefficients.

$$(\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z) = (b_1\mathbf{A}_x + b_2\mathbf{B}_x + b_3\mathbf{C}_x, i\mathbf{A}_y + j\mathbf{B}_y + k\mathbf{C}_y, i\mathbf{A}_z + j\mathbf{B}_z + k\mathbf{C}_z) \quad (5.1)$$

Then we can get Eq.5.2, 5.3, and 5.4.

$$\begin{aligned} \mathbf{p}_x &= i\mathbf{A}_x + j\mathbf{B}_x + k\mathbf{C}_x \\ &= i\mathbf{A}_x + j\mathbf{B}_x + (1 - i - j)\mathbf{C}_x \end{aligned} \quad (5.2)$$

$$\begin{aligned} \mathbf{p}_y &= i\mathbf{A}_y + j\mathbf{B}_y + k\mathbf{C}_y \\ &= i\mathbf{A}_y + j\mathbf{B}_y + (1 - i - j)\mathbf{C}_y \end{aligned} \quad (5.3)$$

$$\begin{aligned} \mathbf{p}_z &= i\mathbf{A}_z + j\mathbf{B}_z + k\mathbf{C}_z \\ &= i\mathbf{A}_z + j\mathbf{B}_z + (1 - i - j)\mathbf{C}_z \end{aligned} \quad (5.4)$$

Simultaneous Eq.5.2 and 5.3, we get Eq.5.5.

$$\begin{cases} \mathbf{p}_x - \mathbf{C}_x + j(\mathbf{C}_x - \mathbf{B}_x) = i(\mathbf{A}_x - \mathbf{C}_x) \\ \mathbf{p}_y - \mathbf{C}_y + j(\mathbf{C}_y - \mathbf{B}_y) = i(\mathbf{A}_y - \mathbf{C}_y) \end{cases} \tag{5.5}$$

If $(\mathbf{C}_x - \mathbf{B}_x)(\mathbf{A}_y - \mathbf{C}_y) - (\mathbf{C}_y - \mathbf{B}_y)(\mathbf{A}_x - \mathbf{C}_x) \neq 0$, we have 5.6

$$j = \frac{-(\mathbf{p}_x - \mathbf{C}_x)(\mathbf{A}_y - \mathbf{C}_y) + (\mathbf{p}_y - \mathbf{C}_y)(\mathbf{A}_x - \mathbf{C}_x)}{(\mathbf{C}_x - \mathbf{B}_x)(\mathbf{A}_y - \mathbf{C}_y) - (\mathbf{C}_y - \mathbf{B}_y)(\mathbf{A}_x - \mathbf{C}_x)} \tag{5.6}$$

If $\mathbf{A}_x - \mathbf{C}_x \neq 0$, we can get i according to Eq.5.7.

$$i = \frac{(\mathbf{p}_x - \mathbf{C}_x) + j(\mathbf{C}_x - \mathbf{B}_x)}{(\mathbf{A}_x - \mathbf{C}_x)} \tag{5.7}$$

If $\mathbf{A}_x - \mathbf{C}_x = 0$, but $\mathbf{A}_y - \mathbf{C}_y \neq 0$, we can get i according to Eq.5.8.

$$i = \frac{(\mathbf{p}_y - \mathbf{C}_y) + j(\mathbf{C}_y - \mathbf{B}_y)}{(\mathbf{A}_y - \mathbf{C}_y)} \tag{5.8}$$

Then, we can get k from Eq.5.9.

$$k = 1 - i - j \tag{5.9}$$

If $(\mathbf{C}_x - \mathbf{B}_x)(\mathbf{A}_y - \mathbf{C}_y) - (\mathbf{C}_y - \mathbf{B}_y)(\mathbf{A}_x - \mathbf{C}_x) = 0$, simultaneous Eq.5.2 and 5.4 and we get Eq.5.10.

$$\begin{cases} \mathbf{p}_x - \mathbf{C}_x + j(\mathbf{C}_x - \mathbf{B}_x) = i(\mathbf{A}_x - \mathbf{C}_x) \\ \mathbf{p}_z - \mathbf{C}_z + j(\mathbf{C}_z - \mathbf{B}_z) = i(\mathbf{A}_z - \mathbf{C}_z) \end{cases} \tag{5.10}$$

If $(\mathbf{C}_x - \mathbf{B}_x)(\mathbf{A}_z - \mathbf{C}_z) - (\mathbf{C}_z - \mathbf{B}_z)(\mathbf{A}_x - \mathbf{C}_x) \neq 0$, we can get j from Eq.5.11.

$$j = \frac{(\mathbf{p}_z - \mathbf{C}_z)(\mathbf{A}_x - \mathbf{C}_x) - (\mathbf{p}_x - \mathbf{C}_x)(\mathbf{A}_z - \mathbf{C}_z)}{(\mathbf{C}_x - \mathbf{B}_x)(\mathbf{A}_z - \mathbf{C}_z) - (\mathbf{C}_z - \mathbf{B}_z)(\mathbf{A}_x - \mathbf{C}_x)} \tag{5.11}$$

If $\mathbf{A}_x - \mathbf{C}_x \neq 0$, we can have i from Eq.5.12.

$$i = \frac{(\mathbf{p}_x - \mathbf{C}_x) + j(\mathbf{C}_x - \mathbf{B}_x)}{(\mathbf{A}_x - \mathbf{C}_x)} \tag{5.12}$$

If $\mathbf{A}_x - \mathbf{C}_x = 0$ but $\mathbf{A}_z - \mathbf{C}_z \neq 0$, we can get i from Eq.5.13.

$$i = \frac{(\mathbf{p}_z - \mathbf{C}_z) + j(\mathbf{C}_z - \mathbf{B}_z)}{(\mathbf{A}_z - \mathbf{C}_z)} \tag{5.13}$$

If $(\mathbf{C}_x - \mathbf{B}_x)(\mathbf{A}_y - \mathbf{C}_y) - (\mathbf{C}_y - \mathbf{B}_y)(\mathbf{A}_x - \mathbf{C}_x) = 0$ and $(\mathbf{C}_x - \mathbf{B}_x)(\mathbf{A}_z - \mathbf{C}_z) - (\mathbf{C}_z - \mathbf{B}_z)(\mathbf{A}_x - \mathbf{C}_x) = 0$, we can simultaneous Eq.5.3 and 5.4, we get Eq.5.14.

$$\begin{cases} \mathbf{p}_y - \mathbf{C}_y + j(\mathbf{C}_y - \mathbf{B}_y) = i(\mathbf{A}_y - \mathbf{C}_y) \\ \mathbf{p}_z - \mathbf{C}_z + j(\mathbf{C}_z - \mathbf{B}_z) = i(\mathbf{A}_z - \mathbf{C}_z) \end{cases} \tag{5.14}$$

If $(\mathbf{C}_y - \mathbf{B}_y)(\mathbf{A}_z - \mathbf{C}_z) - (\mathbf{C}_z - \mathbf{B}_z)(\mathbf{A}_y - \mathbf{C}_y) \neq 0$, we can get j from Eq.5.15.

$$j = \frac{(\mathbf{p}_z - \mathbf{C}_z)(\mathbf{A}_y - \mathbf{C}_y) - (\mathbf{p}_y - \mathbf{C}_y)(\mathbf{A}_z - \mathbf{C}_z)}{(\mathbf{C}_y - \mathbf{B}_y)(\mathbf{A}_z - \mathbf{C}_z) - (\mathbf{C}_z - \mathbf{B}_z)(\mathbf{A}_y - \mathbf{C}_y)} \tag{5.15}$$

If $\mathbf{A}_y - \mathbf{C}_y \neq 0$, we have i from Eq.5.16.

$$i = \frac{(\mathbf{p}_y - \mathbf{C}_y) + j(\mathbf{C}_y - \mathbf{B}_y)}{(\mathbf{A}_y - \mathbf{C}_y)} \tag{5.16}$$

If $\mathbf{A}_y - \mathbf{C}_y = 0$ but $\mathbf{A}_z - \mathbf{C}_z \neq 0$, we have i from Eq.5.17.

$$i = \frac{(\mathbf{p}_z - \mathbf{C}_z) + j(\mathbf{C}_z - \mathbf{B}_z)}{(\mathbf{A}_z - \mathbf{C}_z)} \tag{5.17}$$

Until now, we can obtain the interpolation coefficients of projection points in triangle faces. We record coefficients i, j, k as b_1, b_2, b_3. Next section, we will show how to use these interpolation coefficients to calculate the final deformed input model.

5.4 Deformation of the Input Model

Deform the proxy model using the spherical style deformation method with $a = 0$ (only position directions as target directions). We denote the deformed proxy model as X. In Fig.5.1, the green line represents the shape X. $\mathbf{q}'_1, \mathbf{q}'_2$, and $\mathbf{q}'_3$ are deformation positions of $\mathbf{q}_1, \mathbf{q}_2$, and $\mathbf{q}_3$ respectively on X shown in Fig.5.2(b). $\mathbf{q}'_i$ is the deformed position of $\mathbf{q}_i$, and it satisfies the formula $\mathbf{q}'_i = b_1 * \mathbf{q}'_1 + b_2 * \mathbf{q}'_2 + b_3 * \mathbf{q}'_3$.

Interpolate the input model and X, and obtain the final deformation of the input model. The formula is shown as Eq. (5.18).

$$\mathbf{p}'_i = (1 - t) * \mathbf{q}'_i + t * \mathbf{p}_i \qquad (0 \leq t \leq 1) \tag{5.18}$$

$\mathbf{p}'_i$ is the final deformation position of $\mathbf{p}_i$. t is the interpolation coefficient. The domain of t is from 0 to 1. t can be expressed as a function of points on the input model, and users can set different t functions. This method also provides an option for partial deformation of the input model. When applying partial deformation, t values of points near the connection boundaries are preferably equal to 1. Therefore, the connection is smooth (experimental results are shown in section 5.3).

About the deformation of the proxy model method, we usually give large λ in Eq. (4.2), such as $\lambda = 5$, and X is much rounder.

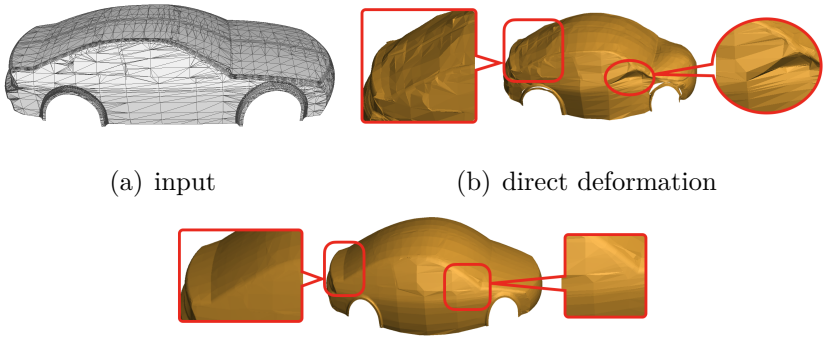
This method is applied to the input models, which are approximately convex models. Projection vectors from points on the input model to corresponding projection points on the proxy model have almost no intersection. If the input model is complex, the projection vectors are not guaranteed with no intersections.

5.5 Results

In this section, we perform the proxy deformation method. The method globally deforms the mesh rounder whose surface consists of some non-regular triangles. We provide various t functions, and partial deformation options. In our experiments, the number of sampling points is smaller than 10k. The models are exported from 3D EXPORT.

Figure 5.3(a) shows the input car model (© basedonmythoughts under CC BY), consisting of some obtuse triangles. When we directly applied spherical style deformation method with position directions as target directions ($a = 0$) to this input model, the result is matte, such as the roof and trunk of the car, shown in Fig.5.3(b). In comparison, our proxy method yields smoother result as shown in Fig.5.3(c). For the approximately convex input model, which consists of obtuse triangles, our proxy method deforms the whole model rounder. Figure 5.4(a)-5.4(c) show the deformation results when the interpolation parameter t is set to 0.8, 0.5, and 0.2, respectively. As the value of t decreases, the surface of the car is rounder, while preserving the local details of the car. Our method allows users to emphasize against an coordinate axis. Figure 5.5 shows the results enlarged against the y- and z-axis, respectively. Because the triangular meshes of the original input car model are not fully connected. Zooming in on the original input car mesh model (Fig.5.3(a)), there are cracks between the dense triangular mesh near the fender and the sparse triangular mesh of the car body. Therefore, we can see that the deformed meshes have cracks in Fig.5.3(b) or holes in Fig.5.4(b), 5.4(c), 5.5(a) and 5.5(b).

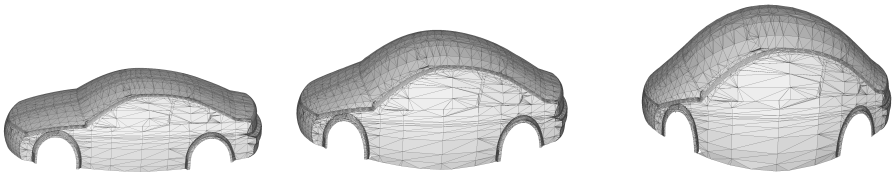
We also deform the stone house model with the proxy method with $t = 0.5$ shown in Fig.5.6. Compared with Fig.4.26(b), the deformation result with the proxy method is better to preserve the shape of stones on the house surface. Therefore, convex hull is better than the sphere as the proxy model. Mean-



(a) input (b) direct deformation

(c) proxy deformation

Figure 5.3: Car deformation comparison

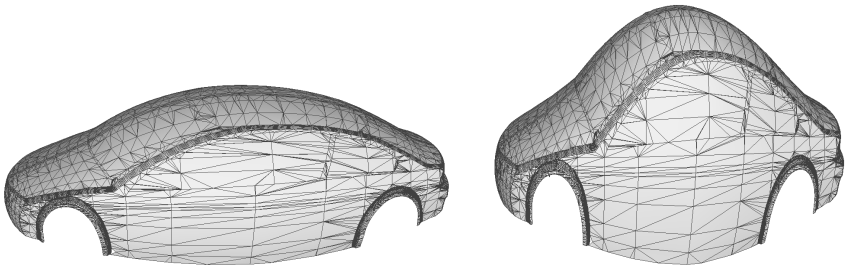


(a) $t = 0.8$ (b) $t = 0.5$ (c) $t = 0.2$

Figure 5.4: Car deformation with different t values

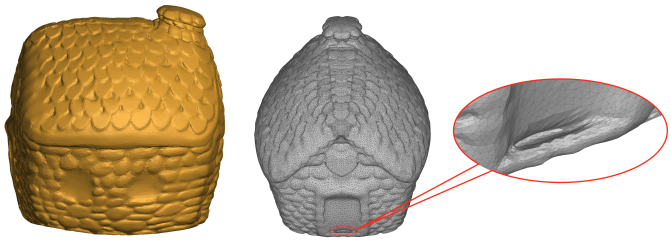
while, the deformation results of the proxy method is affected by projections. In Fig.5.6(b), the points, around the door in the red circle, project down to the proxy model, and deforms the door a little more concave. Users can change the projection standard of shortest distance first to smallest angle first between projection vector and normal vector to alleviate this concave situation. Users can also select these points and reset the projection directions of these points to modify the concave situation.

We take an example: parameter t can be expressed as a function of positions according to an axis. Figure 5.7 shows deformations of different t functions and partial deformations. The parameter t is represented as a function of the vertical axis of the input bottle (© POCTOB under CC BY). We have applied quadratic (a), linear (b), and partially tanh (c) functions, shown in the second row (the vertical axis represents the t value, and the horizontal



(a) enlarge y (b) enlarge z

Figure 5.5: Deformation by enlarging axis



(a) proxy deformation (b) mesh form

Figure 5.6: Stone house deformation of proxy method

axis represents the value of the independent variable with respect to the axis of the bottle.). The projection vectors from points of the bottle to the proxy model are represented by short red lines shown in the first row. In Fig.5.7(a), we deform the whole bottle with quadratic t function. The smaller the t, the rounder the bottle. We also deform bottle partially, such as the part below the red line shown in the first row of Fig.5.7(b), and 5.7(c) with linear and tanh t functions respectively.

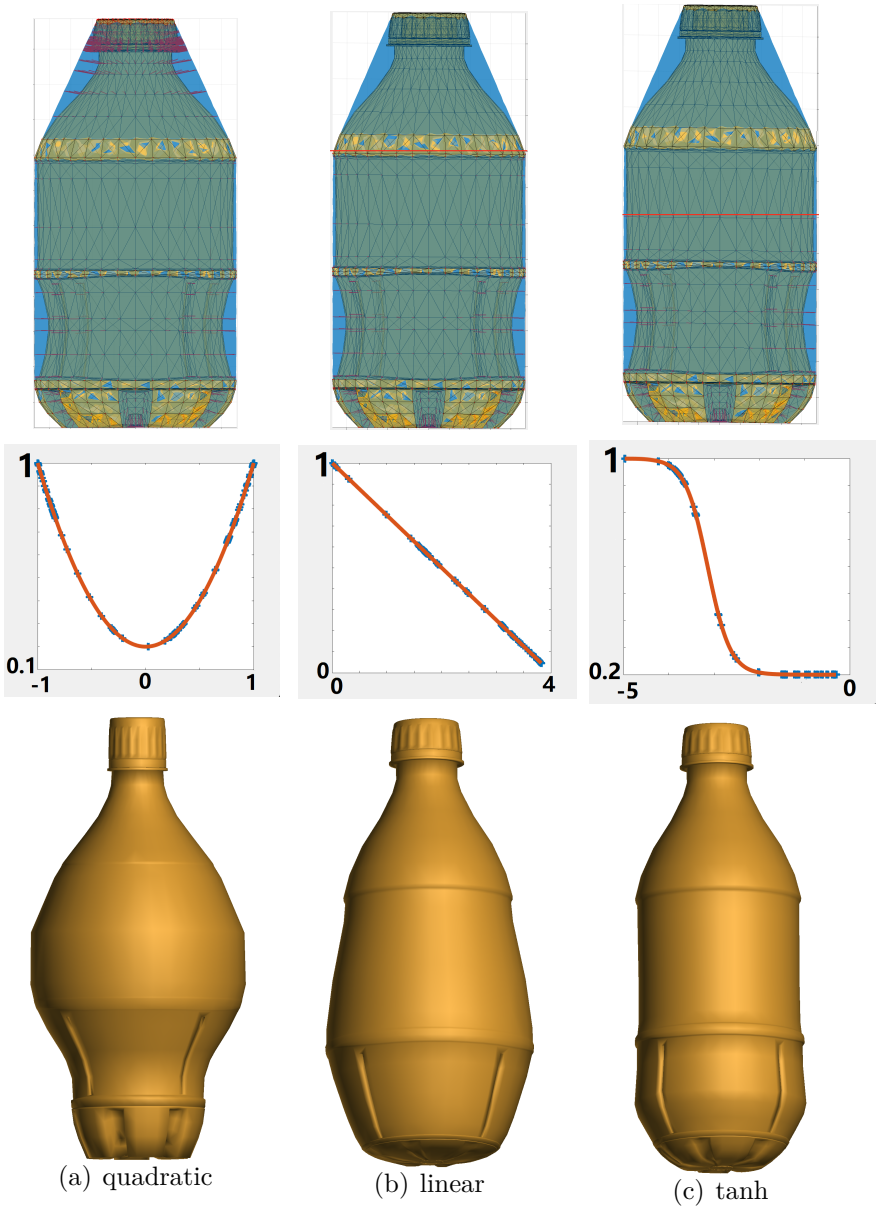
Our method also deforms internal part of the input model. Figure 5.8 shows an example of deforming the head part of the easterbunny (© Witalk73 under CC BY) from two views. We define t functions as linear and quadratic functions, shown in the third row of Fig.5.8(b), and 5.8(c) respectively.

The deformation can be controlled by changing the interpolation t function. When the t values of points near the connection boundaries are equal

to 1, the connection is smooth.

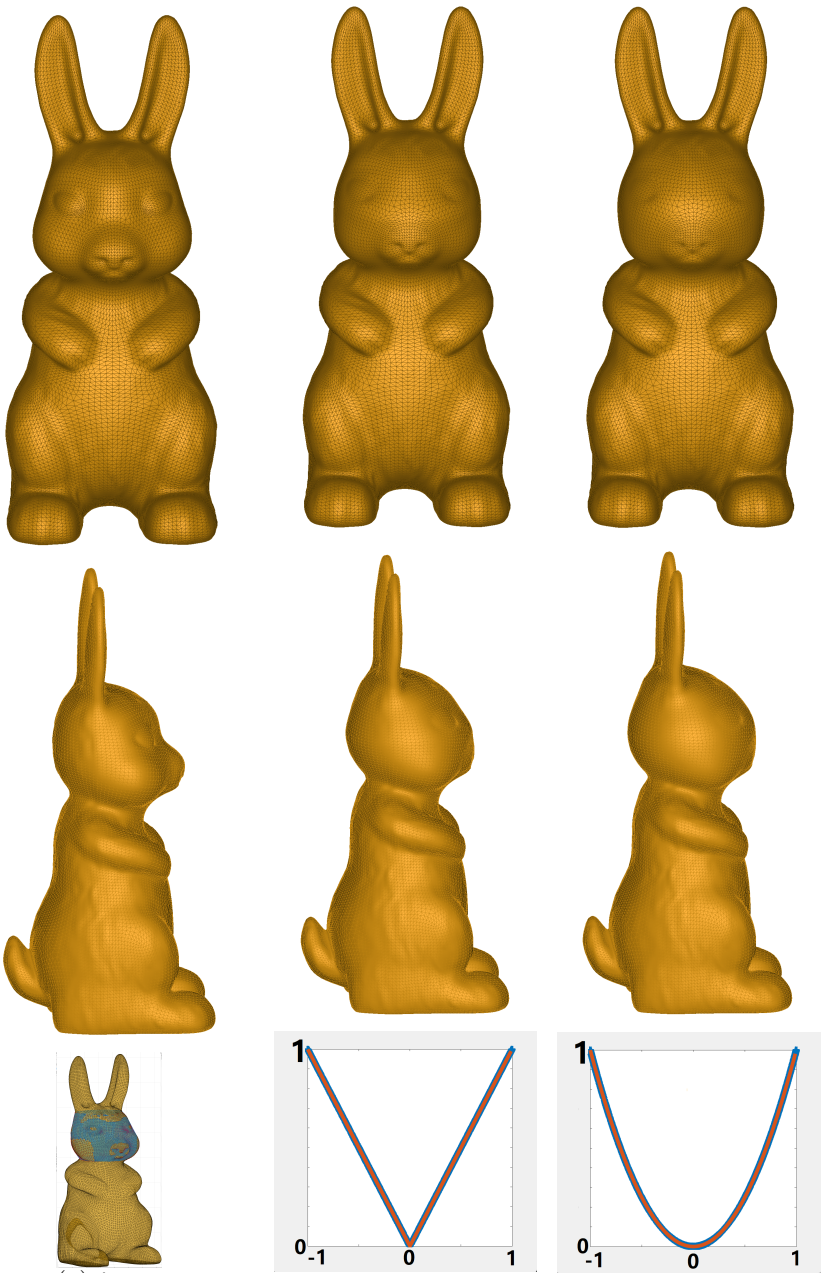
5.6 Limitation of Remeshing

To solve many obtuse triangles of the input model, the deformation method of the proxy model can be an option, which adopts convex hull as proxy model and remeshes the convex hull. However, Ball-Pivoiting algorithm [2] is also limited. When the region around the boundary of the convex hull is very thin, such as the dorsal fin of fish, the proxy model has holes or mesh intersections around the thin boundary area.



(a) quadratic (b) linear (c) tanh

Figure 5.7: Bottle deformation with different t functions



(a) input (b) linear (c) quadratic

Figure 5.8: Easterbunny head deformation

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In our experiment, we use 3D models with the same mesh resolution. In this thesis:

(1) We have proposed spherical style deformation algorithm on single component 3D model. We confirmed that our algorithm can deform surface smooth, rounder, and curve.

(2) For poor mesh quality, we proposed an optional deformation method based on convex hull proxy model as the complementary deformation method for spherical style deformation on single component models.

6.2 Future Work

In this section, we will describe future work in two subsections.

6.2.1 Spherical Style Deformation

In future, we will improve our algorithm from following views.

(1) We will improve the mapping correspondence approach, not only applied on single component model but also components model. More than one sphere centers will be set to improve the mapping correspondence approach. One option is based on the skeleton of the model to set centers. For example, the body component of the bunny corresponds to one sphere center, and the head component of the bunny corresponds to another sphere center.

(2) An interactive spherical style deformation system will be developed. User can set sphere center interactively. At the same time, multi-resolution technique will be applied to speed up the system [15].

(3) In future, we will explore whether there are relationships between spherical surface and other surface shapes.

(4) In future, we will continue to explore the deforamtion scale of every point according to different shapes, when normal vectors are as geometric feature.

6.2.2 Stylization from Shape Analysis

From shape analysis, parametrization approach can stylize various styles of models. However, currently, it is also a difficult research topic. This topic consists of three subtopics.

(1) Topology: when the base model and the style model have different topologies, such as different genuses, parameterization is difficult.

(2) Similar shape recognition: observing successful styled model results, we can conclude that the parts with similar shapes are selected as the corresponding component options. Therefore, similar shape recognition is another subtopic.

(3) Parameterization: the establishment of corresponding parts gives a rough correspondence. Parameterize two complete models to establish point-

to-point correspondence. In the process of parameterization, no mesh flipping and self-intersections are considered.

Bibliography

[1] Marshall W Bern and Paul E Plassmann. Mesh generation. *Handbook of computational geometry*, 38, 2000.

[2] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999.

[3] Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. A simple geometric model for elastic deformations. *ACM transactions on graphics (TOG)*, 29(4):1–6, 2010.

[4] Zhiqin Chen, Vladimir G Kim, Matthew Fisher, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. Decor-gan: 3d shape detailization by conditional refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15740–15749, 2021.

[5] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE transactions on visualization and computer graphics*, 18(6):914–924, 2012.

[6] Noah Duncan, Lap-Fai Yu, Sai-Kit Yeung, and Demetri Terzopoulos. Zoomorphic design. *ACM Transactions on Graphics (TOG)*, 34(4):1–13, 2015.

[7] Yi-Jheng Huang, Wen-Chieh Lin, I-Cheng Yeh, and Tong-Yee Lee. Geometric and textural blending for 3d model stylization. *IEEE transactions on visualization and computer graphics*, 24(2):1114–1126, 2017.

[8] Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. Fast automatic skinning transformations. *ACM Transactions on Graphics (TOG)*, 31(4):1–10, 2012.

[9] Tao Jiang, Kun Qian, Shuang Liu, Jing Wang, Xiaosong Yang, and Jianjun Zhang. Consistent as-similar-as-possible non-isometric surface registration. *The Visual Computer*, 33(6):891–901, 2017.

[10] Wang Jiangrong. Zhihu. `https://zhuanlan.zhihu.com/p/361943207`, Retrieved Aug. 24, 2023. (in Chinese).

[11] Maximilian Kohlbrenner, U Finnendahl, T Djuren, and Marc Alexa. Gauss stylization: Interactive artistic mesh modeling based on preferred surface normals. In *Computer Graphics Forum*, volume 40, pages 33–43. Wiley Online Library, 2021.

[12] Zohar Levi and Craig Gotsman. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE transactions on visualization and computer graphics*, 21(2):264–277, 2014.

[13] Hsueh-Ti Derek Liu and Alec Jacobson. Cubic stylization. *ACM Transactions on Graphics*, 2019.

[14] Hsueh-Ti Derek Liu and Alec Jacobson. Normal-driven spherical shape analogies. In *Computer Graphics Forum*, volume 40, pages 45–55. Wiley Online Library, 2021.

[15] Josiah Manson and Scott Schaefer. Hierarchical deformation of locally rigid meshes. In *Computer Graphics Forum*, volume 30, pages 2387–2396. Wiley Online Library, 2011.

[16] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[17] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33:7198–7211, 2020.

[18] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36, 1993.

[19] Justin Solomon. *Numerical algorithms: methods for computer vision, machine learning, and graphics*. CRC press, 2015.

[20] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[21] Kenshi Takayama, Ryan Schmidt, Karan Singh, Takeo Igarashi, Tamy Boubekeur, and Olga Sorkine. Geobrush: Interactive mesh geometry cloning. In *Computer Graphics Forum*, volume 30, pages 613–622. Wiley Online Library, 2011.

[22] Max Wardetzky, Saurabh Mathur, Felix Kälberer, and Eitan Grinspun. Discrete laplace operators: no free lunch. In *Symposium on Geometry processing*, pages 33–37. Aire-la-Ville, Switzerland, 2007.

List of Publications

- X. Feng, Q. Fang, K. Konno, K. Matsuyama: "An Examination of Rounder Deformation Optimization Combining As-Rigid-As-Possible and Spherical Feature", NICOGRAPH International 2022, pp.56-59, 2022.
- X. Feng, Q. Fang, K. Konno, Z. Zhang, K. Matsuyama: "Spherical Style Deformation on Single Component Models", IEICE Transactions, accepted.