NANYANG TECHNOLOGICAL UNIVERSITY

Real-time Nonlinear Shape interpolation

LIU YAXIN

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN DIGITAL MEDIA TECHNOLOGY

2016

Contents

Abstract

Shape interpolation is one fundamental technique in animation. Conventional ideas for shape interpolation are to interpolate the deformation by separately interpolating the translation and rotation, since the rotation interpolation cannot be treated as the linear interpolation as the translation. Diverse methods have been proposed to solve the problem of non-linear rotation interpolation. Different from traditional ones, a shape interpolation method is proposed in this dissertation to interpolate the translation and rotation simultaneously. This proposed method makes it possible to linearly interpolate translation and meanwhile to nonlinearly interpolate rotation. It is built on the example-driven deformation proposed by Stefan Fröhlich and Mario Botsch in 2011, which incorporates the physics-based deformation into the example-driven deformation and formulates the deformation simply by the energy minimization. The dissertation also reports some experiments to demonstrate the performance of the proposal shape interpolation.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Associate Professor Zheng Jian-Min for his guidance of my Dissertation, for his patience, motivation, enthusiasm, and immense knowledge. He helped me throughout the period of research and writing of the dissertation.

Also, I must acknowledge teachers of Digital Media Technology. I appreciate Assoc. Professor He Ying for his lectures on Computer Graphics and OpenGL Programming. Assoc. Prof Zheng Jianmin’s lectures on 3D modeling and reconstruction, which lay the foundation of my dissertation. And Assoc. Professor Qian Kemao’s 2D & 3D animation course teaches me the basic concepts and principles of shape interpolation. They have helped me understand the concepts and apply them in this dissertation. Furthermore, I appreciate the help from secretary Grace for their help in the School of Computer Science and Engineering.

At last, I want to thank my parents for supporting me in the study of NTU and thank my friends for accompanying me and encouraging me all the time.

Liu Yaxin

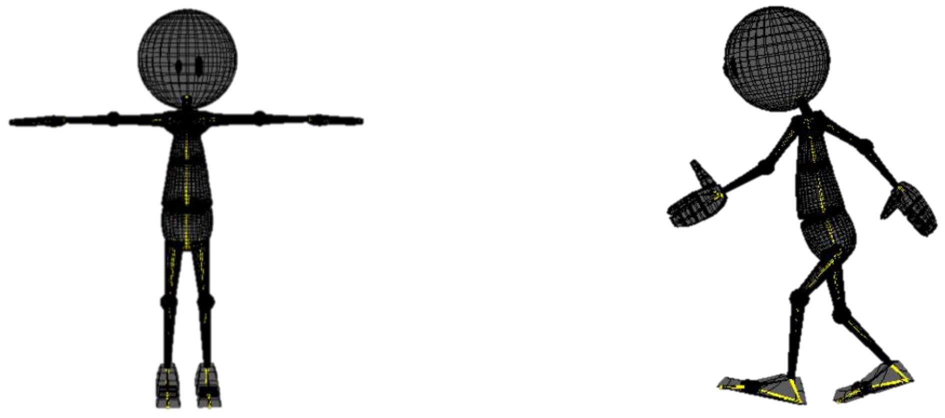
Acronyms

FFD	Free-form Deformation
ROI	Region of Interest

Chapter1. Introduction

1.1 Background

Shape interpolation plays an important role in animation. Generally, animation is divided into three process: modeling, rigging and skinning and animating. Modeling is the process that animators create the characters. Rigging is to build the skeleton for the character and then skinning is to bind the skeleton with the model. For example, animator models a character shown in Fig.1.1a. And rigging is to add bones (the yellow part in the Fig.1.1a) for the character. Skinning connects bones with the model, which enables the model to move with its bones. After the rigging and skinning, animators manipulate correspondent joints and create the character with different posture. Fig.1.1b shows a walking model by manipulating the arms and legs of Fig.1.1a.



a. The model and skeleton b. The walking model

Fig.1.1 The illustration of modeling and rigging/skinning

The final step-animating is a phase for animators to live the character. Key framing is the general technique for animating. Animators set the starting point and ending point of the model, which is called to set the key frames. And then the shape interpolation takes the stage. Shape interpolation fills in the missing frames during two key frames. This makes the character move smoothly as we see in the animation film. Thus, a good approach of shape interpolation is significant for animation.

1.2 Objective and Methodology

With the purpose to interpolate in-between shapes among two key-frames, a shape interpolation method is proposed in the dissertation. The idea is built upon the mesh deformation method proposed by Stefan Fröhlich and Mario Botsch in 2011, which incorporates the physics-based deformation into the example-driven deformation [1]. As for the proposed shape interpolation method, two key-frames are given, denoted as the starting key-frame and the ending key-frame. With the reference to the principle of their example-driven deformation, shapes between two key-frames are obtained.

This shape interpolation method regards the starting key-frame as the undeformed model and the ending key-frame as the example of deformation. By the example-driven deformation, in-between shapes are deformed in a similar way to the ending key-frame. For example, if the ending key-frame is obtained by stretching certain vertices of the undeformed model along x-positive direction, certain vertices of in-between shapes have the similar movement along x-positive direction.

1.3 Organize of The Dissertation

The following chapters are arranged as: in Chapter 2 the related work of shape deformation and shape interpolation are reviewed; in Chapter 3 the framework of example-driven deformation in [1] is presented, and the proposed shape interpolation is put forward; in Chapter 4 a series of experiments on the performance of the mesh deformation and shape interpolation are described; and in Chapter 5 the dissertation is summarized.

Chapter2. Related Work

As shape interpolation methods are based on the principles of shape deformation, shape deformation approaches are firstly reviewed and shape interpolation methods are summarized in the following.

2.1 Shape deformation

Until now, many researches have dedicated on shape deformation. According to different representation of deformable objects, shape deformation is roughly classified into five categories: free-form deformation, multi-scale deformation, differential domain deformation, physics-based deformation, and example-driven deformation.

Firstly, Freeform Deformation (FFD) approximates a space to represent a model, which embeds the model. Users deform the model by indirectly deforming the approximate space. The space is made up of either the regular primitives (e.g. box, lattice) or mathematical related ones (e.g. B-spline control points). In 1986, it is originally formulated by Sederberg and Parry in [2], where the deformable object is represented by the tensor product Bernstein polynomial. The deformable object is embedded in a 3D parallelepiped lattice formed by the control points of the polynomial. Generally speaking, four steps are included in FFD: ①define a parametric volume, including parametric space and lattice; ②map object into parametric space of volume; ③modify shape of volume through editable control points in lattice; ④deform the object embedded in volume. Obviously, FFD has two advantages: ①deformations generated by FFD is independent of the complexity of deformable objects; ②the interaction with users is intuitive. Figure 2.1 shows an example of FFD in [3].
Furthermore, many extensions about FFD have been proposed. In 1989, Greissmair and Purgathofer in [4] put forward B-spline FFD by extending the original formulation to trivariate B-splines. Later in 1998, Jieqing Feng and Pheng-Ann Heng in [5] improved B-spline FFD for singular points and sampling problems through functional composition

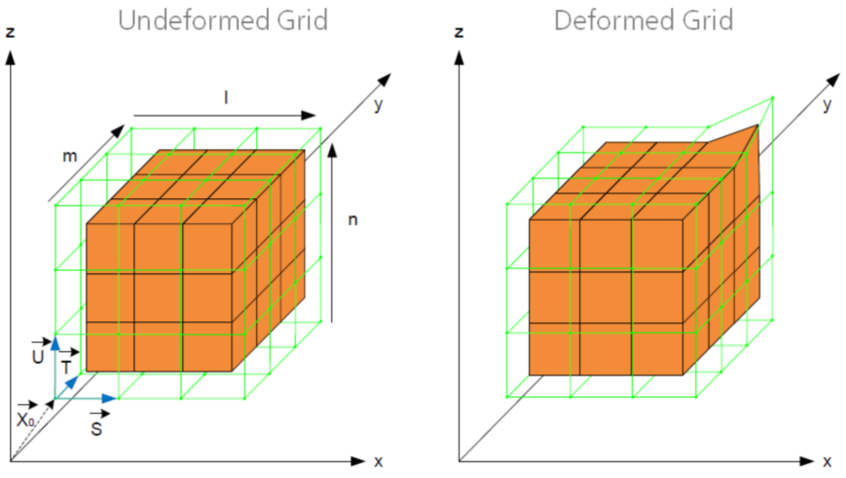


Fig. 2.1 An example of FFD [3]

and shifting operator. In 1990, Coquillart in [6] and [7] extended the FFD from original 3D parallelepiped lattice to non-parallelepiped 3D lattice. Also, in 1998, Singh and Fiume in [8] borrowed the idea of FFD to bind the deformable objects with curves drawn on its surface. Thus, the mesh deformation is by the manipulation of the curves.

Secondly, as for multi-scale deformation, the original idea came from the multi-resolution analysis on wavelet. It was first formulated in [9] by Lounsbery, DeRose, and Warren in 1997. An input model is represented by a base mesh and a sequence of detail coefficients. The base mesh turns into the input model after a series of subdivision. And the detail coefficients are the indicators of this subdivision process. By generating the hierarchy of the input mesh, users are able to change the mesh at a coarser resolution without changing the details. Multi-resolution methods better preserve the details during deformation. As shown in Fig.2.2 of [10], the input model is S and it is decomposed into base mesh B and detail information. Then users edit the base mesh B to B'. After adding the detail information to B', the result of deformation is S'.

One typical problem of the multi-scale representation is the subdivision connectivity of the input model due to the subdivision process. A necessary preprocessing technique in [11] for the subdivision connectivity is remeshing. Another problem is that it is difficult to choose the vertices correspondent to features they want when users editing the mesh at coarser resolution. Concerning these problems, a smoothing strategy was proposed by Kobbelt and his colleagues in [12]. The vertex displacements between the smoothed and original surface are the detail coefficients of the model. In 2000, Kobbelt proposed a dynamic mesh representation in [13]. The global hierarchical structure of the model is replaced by representing the details with the difference between individual mesh. Also, the detail coefficients function is set up by the normal displacement of vertices in one level to the next level. One limitation of this method is the application for complex model, either complex topology or complex geometry, since this requires more complex hierarchical structure. Another limitation is the lack of information about the neighbors during deformation

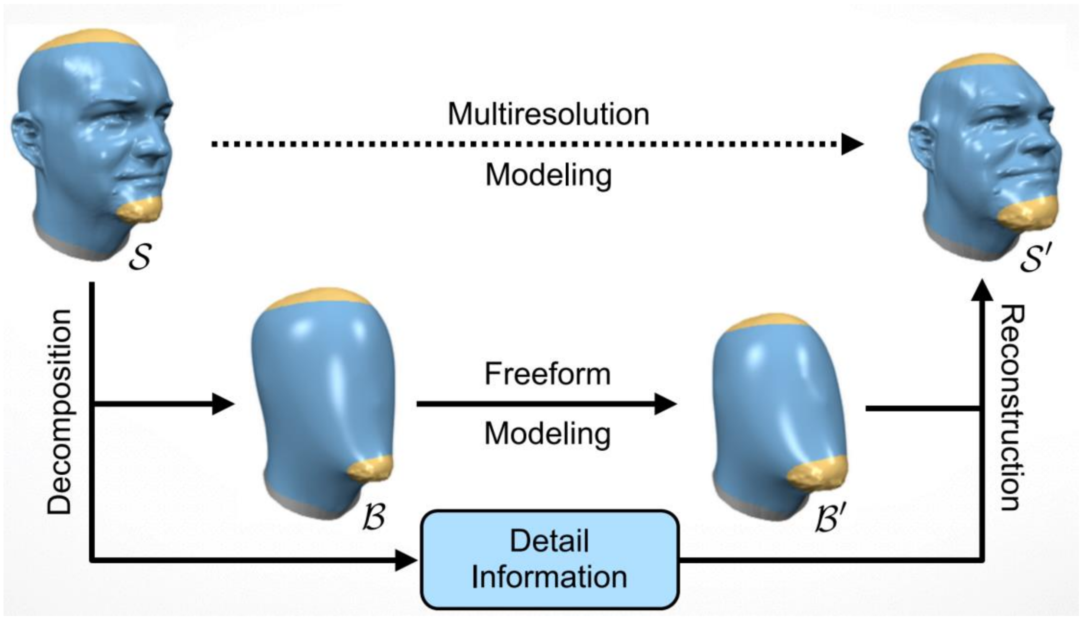


Fig. 2.2 Multiresolution deformation [10]

Thirdly, the model represented by Laplacian coordinates or differential coordinate is also better at detail preserving. The model in this scheme is represented by the weighted sum of differences between the vertices and its neighbors. Laplacian coordinate was first employed by Lipman for the mesh deformation in [14]. Different from other methods mentioned above using the Cartesian coordinate, this approach applies the local relationship among a vertex and its neighbors. This is why Laplacian coordinate representation preserves the mesh's details. The principle of Laplacian coordinate is talked about in the following.

The Laplacian coordinate δ^i of vertex $v^i$is formulated as $\delta_i = L(v_i)$, where L is the Laplacian operator, shown in (1):

$$L(v_i) = v_i - \sum_{j \in N_i} \omega_j v_j \tag{1}$$

where ω_j is the weight, showing the relationship between v_i and v_j, N_i is the neighbor of the vertex v_i. Because of its application to non-uniform tessellation, cotangent weight is often used for computation. So the Laplacian coordinate of the model is denoted as (2):

$$\boldsymbol{LV} = \boldsymbol{\delta} \tag{2}$$

where the $\boldsymbol{L}$ is the Laplacian matrix, and $\boldsymbol{V}$ is the vector of Cartesian coordinate and $\boldsymbol{\delta}$ is the vector of Laplacian coordinate. Getting the Laplacian coordinate of the model, the deformation is formulated as energy minimization problem, shown in (3):

$$\underset{V}{min} \|\boldsymbol{LV} - b(\boldsymbol{V}, \boldsymbol{\delta}^0)\|$$

$$\text{Constraints: } C(V) = 0 \tag{3}$$

where $\boldsymbol{\delta}^0$ is the Laplacian coordinate of the original model, and $b(\mathbf{V}, \boldsymbol{\delta}^0)$ is the function to compute the Laplacian coordinate with the reference to the original reference after the user drags vertices to new positions. And *C(V)* is the constraints of the model, like the volume constraints or the skeleton constraints. In [15], (3) is solved by linear or nonlinear optimization under the constraints *C(V).* The solution of (3) is the Laplacian coordinates after transformation. The final deformation results will be obtained by mesh reconstruction with Poisson equation.

Geodesic interpolation is a linear method. As shown in Fig.2.3 in [16], users first choose one boundary condition *BC* (Fig.2.3 a) and then transform the boundary condition to *BC'* (Fig.2.3 b). With the reference to the transformation between *BC* and *BC'*, vertices in the boundary condition are able to reposition and these vertices are called the constraint vertices. In order to compute the rest of the vertices, the strength field is computed based on the geodesic distances from these constraint vertices. Then the transformation at each vertex is reproduced with the strength field. The differential coordinates are obtained (Fig.2.3 c). Finally, in [16] the Poison equation is used to reconstruct meshes (Fig.2.3 d). Besides, Harmonic Propagation is proposed in [17] to improve the Geodesic interpolation's unnatural deformation of mesh with protruding features. A scalar field is computed to indicate the propagation of transformation, which sets the source and the sink of the propagation. This technique in [17] causes the problem of different deformation with different source-sink setting.

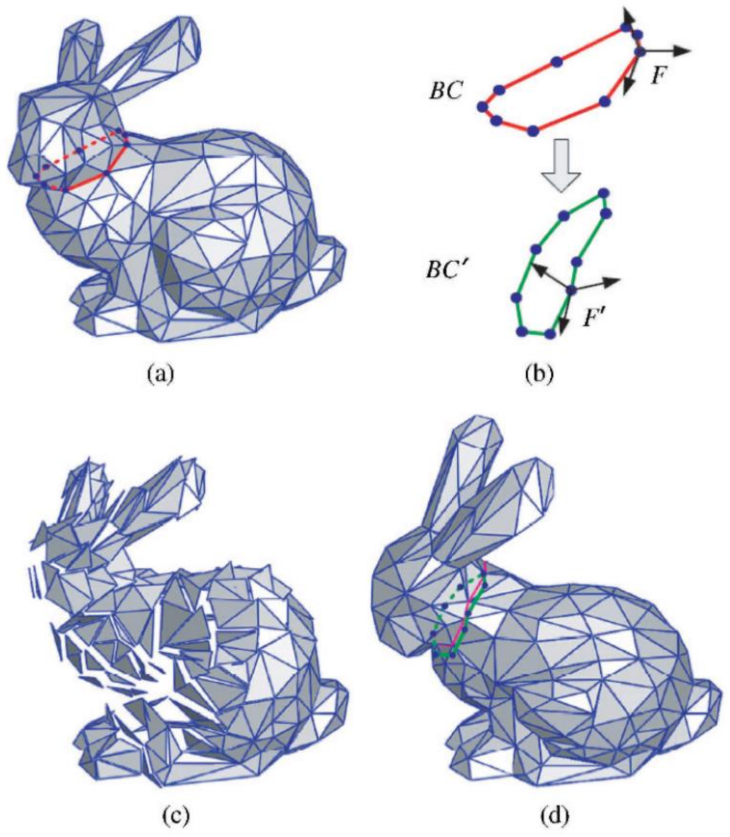


Fig. 2.3 Geodesic interpolation [16]

The basic idea of non-linear methods is to update the vertex coordinates and Laplacian coordinates at each iteration until convergence. Gauss-Newton method is employed. According to (3), the residual function is $f(\boldsymbol{V}) = \boldsymbol{LV} - b(\boldsymbol{V}, \boldsymbol{\delta}^0)$. At iteration k, the $f(\boldsymbol{V}^{\boldsymbol{k+1}}) = f(\boldsymbol{V}^{\boldsymbol{k}}) + (\boldsymbol{L} - \boldsymbol{J}_{\boldsymbol{b}})h$, where h is the iteration step, $\boldsymbol{J}_{\boldsymbol{b}}$ is the Jacobian Matrix of $b(\boldsymbol{V}, \boldsymbol{\delta}^0)$. Since the $b(\boldsymbol{V}, \boldsymbol{\delta}^0)$ is hard to compute analytically, so is $\boldsymbol{J}_{\boldsymbol{b}}$. As the norm of $\boldsymbol{J}_{\boldsymbol{b}}$ is much less than the one of $\boldsymbol{L}$, the omission of $\boldsymbol{J}_{\boldsymbol{b}}$ have less effect on the final convergence. Thus, the function of each iteration is simplified to (4):

$$f(\boldsymbol{V}^{\boldsymbol{k+1}}) = f(\boldsymbol{V}^{\boldsymbol{k}}) + \boldsymbol{L}h = \boldsymbol{LV}^{\boldsymbol{k}} - b(\boldsymbol{V}^{\boldsymbol{k}}, \boldsymbol{\delta}^0) + \boldsymbol{L}h = \boldsymbol{LV}^{\boldsymbol{k+1}} - b(\boldsymbol{V}^{\boldsymbol{k}}, \boldsymbol{\delta}^0) \qquad (4)$$

The constraints of this optimization in [18] and [19]is the boundary condition with updated vertex coordinate. The dual Laplacian coordinates method in [20], with the same spirit as Gauss-Newton method, proposes to update the vertex coordinate with dual mesh in case of the tangential drift effect. The limitation of mesh deformation approaches based on the Laplacian coordinate is translation-insensitivity, since Laplacian coordinate describes the relationship among a vertex and its neighbors. Additionally, the rotation is also the challenge due to the reconstruction process. Lipman et al. in [21] proposed to rotate the Laplacian coordinates according to the displacement estimates of the surface orientation; while Sorkine in [14] employed the linearization of rotation in order to align with the reconstruction process.

Fourthly, physics-based representation, also called shell-based representation, applies the physics theory into model deformation. In 1991, George Celnikera and Dave Gossardb in [22] formulated the mesh deformation as (5) with the assumption that energy is stored in each shape. This align with the elasticity theory. For example, pressing the spring to different length, it stores different energy.

$$E_{deformation} = \int_{\sigma} (\alpha\ stretch + \beta\ bending) d\sigma \tag{5}$$

In this spirit, the shape deformation is to stretch and bend with minimizing the $E_{deformation}$ under the geometric constraints by users. One assumption for this idea is that the shape resists stretching and bending. The stretching weighted term α and bending weighted term β in (5) denotes the amount of resistance of the shape to stretching and bending. [22] explains why it is possible to deform the shape while minimizing the energy (5): it is like the balance situation in our real life. For example, the bottle is on the table statically. For the bottle staying on the table, this static staying is the balance state. Now someone pushes the bottle, and the bottle might fall down. If, in the meantime, another force identically opposed to the pushing force is provided, the total external force would be zero. So the bottle will keep on the table statically as the beginning. Analogous to the mesh deformation, the pushing force is the analogy to constraints or user inputs; the opposed force is the deformed shape; and the process to provide the opposed force is the energy minimization. So users input the deformation of some primitives, it is the process to put external forces to the shape. Then the shape is deformed to balance the external forces, since different shapes store different amount of energy.

How does the shape deform? It results from the nature of the shape. Due to the resistance to stretching, curves tend to minimize the length; and surfaces tend to minimize the area when users stretch them. With the factor of resistance to bending, local bending distributes to a large region. The final effect is the product of realistic and smooth deformation [22]. Similarly, the energy of the shape deformation is formulated as (6) with the first and second fundamental forms in [23] and [24], called shell-based energy:

$$E_{shell}(S') = \int_{\Omega} k_s \|I' - I\|_F^2 + k_b \left\| \Pi' - \Pi \right\|_F^2 dudv \tag{6}$$

where k_s and k_b are used to denote the resistance to stretching and bending. In order to simplify (6), in [22] and [25], with the purpose of less computation, the first-order partial derivative and the second-order partial derivatives of the displacement function d take the place of the first and second fundamental forms, and the stretching and bending energy are linearized into (7). Based on (6), two physics-based models are built: membrane model and thin-plate model, formulated as (8) and (9):

$$E_{shell}(d) = \int_{\Omega} k_s(\|d_u\|^2 + \|d_v\|^2) + k_b(\|d_{uu}\|^2 + 2\|d_{uv}\|^2 + \|d_{vv}\|^2)dudv \tag{7}$$

$$E_{memb}(p) = \int_{\Omega} (\|p_u\|^2 + \|p_v\|^2)dudv \tag{8}$$

$$E_{plate}(p) = \int_{\Omega} (\|p_{uu}\|^2 + 2\|p_{uv}\|^2 + \|p_{vv}\|^2)dudv \tag{9}$$

From (8), it is seen that the bending energy is not taken into consideration. So this model is applicable to material like rubber which mainly resist the deformation of stretching and shearing. And shown in (9), the omission of stretching energy means that the model is suitable to materials with the significant deformations in bending. Fig.2.4 shows the deformation caused by membrane model and thin-plate model and shell-based model [24]. Fig.2.4a is the undeformed region; the grey region is the fixed region; the blue region is the ROI and the yellow region is the handle position. Fig.2.4b is the deformation with the membrane model, where k_s=1 and k_b=0. Fig.2.4c is with the thin-plate model, where k_s=0 and k_b=1. And Fig.2.4d is with the shell-based model, where k_s=1 and k_b=10. One problem with physics-based deformation is that the linearization of the bending and stretching energy have some unnatural deformations or artifacts with large rotational deformations, due to the nonlinear transformation of rotation.

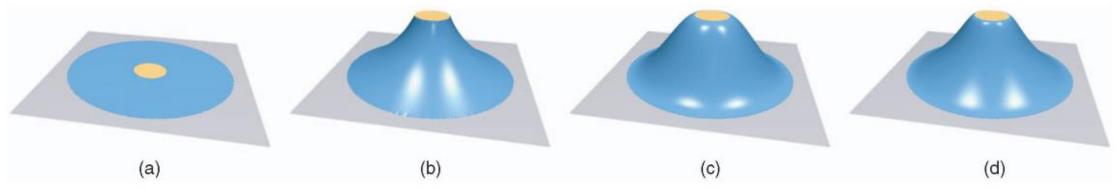


Fig.2.4 Deformation with membrane (b)/ thin-plate (c)/ shell-based model (d) [24]

The *last* one is example-driven deformation. This applies examples as the indicator of the class of the deformation which guide the mesh deformation. That is to say, the model is represented as the weighted sum of examples. Traditional example-driven deformation is the skeleton-based deformation. While, in 2005, Robert W. Sumner, Matthias Zwicker, Craig Gotsman and Jovan Popovic in [26] proposed the mesh-based example-driven deformation, since the mesh-based deformation enables the user-interface where users manipulate several mesh vertices and the entire model is deformed. In their work, they represent each example with a feature vector. The feature vector encodes the affine transformation of the individual triangles in the example poses undergoing from a reference one. Thus, the deformation is to find the closest feature vector to the user constraints in the span of feature space of the example poses. The final mesh is deformed with the closest feature vector. Key technique is how to span the feature space. In [26], both linear and nonlinear method are proposed. One limitation of this approach is that unnatural deformation is possible when the user constraints do not fit in the feature space.

2.2 Shape Interpolation

As for obtaining the in-between shapes while the source shape turns into the target shape, linear interpolation is the simplest way to get results. But undesirable in-between shapes may be produced. To solve this problem, several researches have been done. One is to interpolate the local affine transformations, which is to determine the affine transformation of each pair of primitives instead of the global transformation. In 2000, Marc Alexa, Daniel Cohen-Or and Tel David Levin in [27] proposed an object-space interpolation, which is based on the primitives of the model, such as polygon or triangles. At the beginning, they determined the local affine transformation of the constraint boundary and the interior of the shape from the source model to the target model. Then the local affine transformation, noted as A, is decomposed into rotation and translation (stretching). The rotation and stretching is interpolated with time t. Another transformation matrix is built upon the interpolation of vertices' location, noted as B. Finally, the error function minimizes the quadric difference between A and B. With the local transformation matrix solved, the in-between shapes are determined [27].

In addition, the local interpolation of differential gradient is proposed. In 2005, Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao in [28] put forward the gradient-based interpolation with Poisson equation. At first, the local affine transformation is computed and decomposed. In-between transformations are built with time t. Then, instead of determining the local transformations, the gradient coordinate is interpolated with time t. After that, Poisson equation is used to reconstruct the intermediate shapes with the gradient coordinates at time t [28]. Similar to differential coordinate interpolation, relative coordinate is applied for interpolation. Martin Kilian, Niloy J. Mitra and Helmut Pottmann in [29] employed the geodesics, but the method is too complex. Approaches in [30] proposed by Isaac Chao, Ulrich Pinkall, Patrick Sanan, Peter Schröder in 2010 is applicable only for solid models.

Aside from the local affine transformation and differential gradient, edge length and dihedral angles, as the intrinsic parameters of a polygon or triangle, are used for interpolation. It was first proposed by Sederberg and Greenwood in 1992 with a physical based morphing technique in [31]. Their method interpolates edge length and angles between adjacent edges rather than vertex location. But the optimization of the deformation with edge length and dihedral angles is a problem. With this problem, Tim

Winkler, Jens Drieseberg and Marc Alexa Kai Hormann proposed the interpolation of edge length and dihedral angles based on the hierarchical structure of the mesh in [32]. They interpolate the edge length and dihedral angle at the level of adjacent triangles; and then multi-registration method is employed to combine them with the hierarchical decomposition of the mesh into larger patch. At the bottom of the hierarchy, a single triangle is interpolated linearly as (10):

$$\begin{aligned} a_t &= (1-t)a_0 + ta_1 \\ b_t &= (1-t)b_0 + tb_1 \\ c_t &= (1-t)c_0 + tc_1 \end{aligned} \tag{10}$$

where a_0, b_0, c_0 is the edge length of triangle from the source model, and a_1, b_1, c_1 is the edge length of the corresponding triangle from the target model. At the next coarser level, the adjacent triangles sharing one common edge are considered. The dihedral angle is added to interpolate as (11):

$$\alpha_t = (1-t)\alpha_0 + t\alpha_1 \tag{11}$$

where α_0 is the dihedral angle of the adjacent triangles from the source model, and α_1 is the dihedral angle of the adjacent triangles from the target model. And on the next coarser level, the adjacent triangles are combined and aligned with the multi-registration method.

Chapter3. Shape interpolation based on Example-Driven Deformation

In this chapter, the principle and framework of the shape interpolation is presented. It is built on the example-driven deformation by Stefan Fröhlich and Mario Botsch in 2011 [1]. The following part is organized as: firstly, the example-driven deformation approach is stated; and then, the shape interpolation based on this example-driven deformation is set forth.

3.1 Example-Driven deformation

As mentioned in Chapter 2, the example-driven deformation might produce unnatural results, since users' constraints may not be within the span of the feature space. Stefan Fröhlich and Mario Botsch proposed an alternative to solve this problem by incorporating the physics-based deformation. Physics-based deformation applies the elastic theory to simulate the deformation of the elastic model. Theoretically, it makes the meaningful deformation; while, practically, it is time-consuming for complex models due to the lack of semantic deformation. Considering the limitations of these two principles, combination of two methods might complement the drawbacks of both. The example-driven method provides the semantic deformations for physics-based deformation, which improves the performance of physics-based deformation with complex models. On the other hand, physics-based principle produces natural results when users' constraints do not fall in the feature space.

3.1.1 Framework of Example-driven Deformation

The framework of example-driven deformation involves two parts: the mesh interpolation and mesh deformation. Mesh interpolation is to interpolate given examples so as to match users' constraints as much as possible; and mesh deformation is to deform the interpolated shape to satisfy users' constraints as much as possible. The following parts elaborate the mesh interpolation at first and then the mesh deformation.

3.1.1.1 Mesh Interpolation

The interpolation method is following the idea of Winkler et al. in [32]. The wedge, that consists of two triangles sharing a common edge, is the primitives of the interpolation. Edge length and dihedral angles of the wedge are used to interpolate. Besides, the volume is also introduced for interpolation, since for elastic objects the volume is preserved globally during the deformation. Given the undeformed model *M* which users ask to deform, and k example poses $M_1, M_2, \dots, M_k$, edge length, dihedral angles and volumes of the deformed model are interpolated with input examples as (12), (13), (14):

$$l_e^* = L_e + \sum_{i=1}^{k} \alpha_i (L_e^{(i)} - L_e) \tag{12}$$

$$\theta_e^* = \Theta_e + \sum_{i=1}^{k} \alpha_i (\Theta_e^{(i)} - \Theta_e) \tag{13}$$

$$v^* = V + \sum_{i=1}^{k} \alpha_i (V^{(i)} - V) \tag{14}$$

where l_e^*, θ_e^* and v^* is the edge length, dihedral angle and volume of deformed model; L_e, Θ_e and V is the value of source model or undeformed model *M*; $L_e^{(i)}$, $\Theta_e^{(i)}$ and $V^{(i)}$ is the value of the example pose M_i; and α_i is the interpolation weight of example pose *i*.

3.1.1.2 Mesh Deformation

Physics-based mesh deformation is applied. Stefan Fröhlich and Mario Botsch employed the thin-shell model for model representation. Thin-shell is a kind of thin and flexible structure having high ratio of width to thickness [33], and is very different from its counterparts thin-plate or membrane model. For one thing, without deformation, the thin-shell model is a naturally curved model, compare to the flat shape of the thin-plate and membrane. For another thing, the thin-plate model only involves the bending energy; and the membrane model only considers the effect of stretching energy. So the thin-plate model is usually applied on cloth simulation, and the membrane model is often used for something like rubber. The thin-shell model is affected by both stretching and bending energy, formulated in (15), therefore, thin-shell is an appropriate model to simulate characters in the animation [33].

Thin-shell model, proposed by Eitan Grinspun and Anil N. Hirani et al in [33], is formulated as the non-linear membrane energy and bending or flexural energy in a discretized form as (16) and (17):

$$E = k_M E_M + k_B E_B \tag{15}$$

$$E_M = k_L \cdot \left(\textstyle\sum_e (1 - \|e\|/\|e_0\|)^2 \|e_0\|\right) + k_A \cdot \left(\textstyle\sum_A (1 - \|A\|/\|A_0\|)^2 \|A_0\|\right) \tag{16}$$

$$E_B = \textstyle\sum_e \left(\theta_e - \theta_{e_0}\right)^2 \|e_0\|/h_{e0} \tag{17}$$

where e_0, A_0 and θ_{e_0} is the edge length, area and dihedral angle of the undeformed model, e, A, and θ_e is the value of deformed model, h_{e0} is a third of the average heights of adjacent triangles sharing the edge *e*. Different material is able to model by setting the membrane and bending stiffness k_M and k_B. Membrane energy as (16) formulates the stretching energy by measuring the local changes in area and edge length, which is the first fundamental form of the model. Bending energy is measured by the difference of mean curvature between the deformed and undeformed surfaces, as (17). The most important advantage of the thin-shell model in [33] is that the energy function is discretized with geometric primitives - wedge, which matches the mesh interpolation method in Chapter 3.1.1.1.

Analogous to the thin-shell model in [33], the thin-shell model proposed by Stefan Fröhlich and Mario Botsch is formulated as (18). Besides the stretching energy and bending energy, the global volume presentation is added. Changes of mesh area do not take into account, because the local area change have little influence on the final deformation. Stretching energy and bending energy is both computed by the squared difference between the current edge length/ dihedral angle and the edge length/ dihedral angle of source model, as (19) and (20). The volume preservation term is measured by the deviation of current volume from the source model.

$$E = \lambda E_s + \mu E_b + v E_v \tag{18}$$

$$E_s = \frac{1}{2} \textstyle\sum_{e \epsilon \varepsilon} (l_e - l_e^*)^2 \frac{1}{L_e^2} \tag{19}$$

$$E_b = \frac{1}{2} \textstyle\sum_{e \epsilon \varepsilon} (\theta_e - \theta_e^*)^2 \frac{L_e^2}{A_e} \tag{20}$$

$$E_v = \frac{1}{2}(v - v^*)^2 \frac{1}{V^2} \tag{21}$$

where l_e, θ_e and v is the edge length, dihedral angle and volume of the current model; A_e is the sum of the areas of adjacent triangles; and parameters λ, μ and υ is the stiffness. The volume preservation here is just a semantic measurement, it has no strict physical support. The global volume of the model is computed as (22). It is the sum of the local tetrahedron volume which consists of one face of the input model and the origin point (0,0,0) of the coordinate system. The tetrahedron volume is computed by the scalar triple product.

$$v = \frac{1}{6}\sum_{f_{ijk}\in F}(\boldsymbol{x_i} \times \boldsymbol{x_j}) \cdot \boldsymbol{x_k} \tag{22}$$

where $\boldsymbol{x_i}$, $\boldsymbol{x_j}$ and $\boldsymbol{x_k}$ is the coordinates of three vertices from one triangle mesh of the input model.

3.1.2 Formulation and Implementation

The edge length and dihedral angle of the current model is obtained by mesh interpolation in Chapter 3.1.1.1; and the thin-shell model is used to optimize the interpolated model. The deformed model is the solution of the energy minimization. Combining the mesh interpolation (12), (13), (14) and thin-shell based deformation, the stretching energy and bending energy of the example-driven deformation is formulated as (23). Therefore, mesh interpolation and mesh deformation of example-driven deformation is denoted in one formula. The final deformation is obtained by minimizing the energy.

$$E_s = \frac{1}{2}\sum_{e\epsilon\varepsilon}\{l_e - [L_e + \sum_{i=1}^{k}\alpha_i\left(L_e^{(i)} - L_e\right)]\}^2 \frac{1}{L_e^2}$$

$$E_b = \frac{1}{2}\sum_{e\epsilon\varepsilon}\{\theta_e - [\Theta_e + \sum_{i=1}^{k}\alpha_i(\Theta_e^{(i)} - \Theta_e)]\}^2 \frac{L_e^2}{A_e}$$

$$E_v = \frac{1}{2}(v - v^*)\frac{1}{V} \tag{23}$$

In their work, this energy minimization is considered as the nonlinear least-square problem. The Gauss-Newton iteration method is employed to solve the energy minimization. In the following part, the implementation is stated.

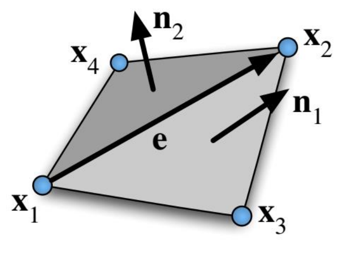
Firstly, the energy minimization problem is formulated: the input model has *n* unconstrained vertices V_1,…, V_n, *m* unconstrained edges and *k* example poses. In [1], the residual function ***f*** is set up as (24):

$$\boldsymbol{f}\text{:}\quad \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \\ y_n \\ z_n \\ \alpha_1 \\ \cdot \\ \cdot \\ \cdot \\ \alpha_k \end{bmatrix} \rightarrow \begin{bmatrix} w_{s,1}(l_1 - l_1^*) \\ \cdot \\ \cdot \\ \cdot \\ w_{s,m}(l_m - l_m^*) \\ w_{b,1}(\theta_1 - \theta_1^*) \\ \cdot \\ \cdot \\ \cdot \\ w_{b,m}(\theta_m - \theta_m^*) \\ w_v(v - v^*) \end{bmatrix} \tag{24}$$

Energy function (18) in [1] changes to $\boldsymbol{E} = \frac{1}{2}\boldsymbol{f}^T\boldsymbol{f}$, where x_i, y_i and z_i is the coordinates of unconstraint vertex V_i; $w_{s,e}$, $w_{b,e}$ and w_v is the weighting factors, computed as (25):

$$w_{s,e} = \sqrt{\lambda \frac{1}{L_e^2}}, \quad w_{b,e} = \sqrt{\mu \frac{L_e^2}{A_e}}, \quad w_v = \sqrt{v \frac{1}{V^2}} \tag{25}$$

Secondly, computation of Jacobian matrix of the residual function ***f*** is based on the method in [34]. The size of the Jacobian matrix is (2*m*+1)×(3*n*+*k*), where *k* is the number of examples. Each row of Jacobian matrix contains certain number of non-zeros: for the edge length constraints, each row contains no more than 6+*k* non-zeros; for the dihedral angles constraints, no more than 12+*k* non-zeros are contained; and for the volume constraints, the row is filled. As shown in the figure below, it is the wedge made up of two adjacent triangles, $\boldsymbol{x_1}$, $\boldsymbol{x_2}$, $\boldsymbol{x_3}$ and $\boldsymbol{x_4}$ are vectors representing the four vertices of the wedge, and $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ are the normal vector of two triangles.



$\mathbf{e} = \boldsymbol{x_2} - \boldsymbol{x_1}$

$\boldsymbol{n_1} = (\boldsymbol{x_3} - \boldsymbol{x_2}) \times (\boldsymbol{x_3} - \boldsymbol{x_1})$

$\boldsymbol{n_2} = (\boldsymbol{x_4} - \boldsymbol{x_1}) \times (\boldsymbol{x_4} - \boldsymbol{x_2})$

The derivatives of the edge length constraints of edge **e** are shown as (26):

$$\frac{\partial}{\partial \boldsymbol{x_1}} w_{s,e}(\|\boldsymbol{e}\| - l^*) = w_{s,e}\frac{-\boldsymbol{e}}{\|\boldsymbol{e}\|}, \qquad \frac{\partial}{\partial \boldsymbol{x_2}} w_{s,e}(\|\boldsymbol{e}\| - l^*) = w_{s,e}\frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \tag{26}$$

The derivatives of dihedral angle constraints are shown as (27):

$$\frac{\partial}{\partial \boldsymbol{x_1}} w_{b,e}(\theta - \theta^*) = w_{b,e}[\frac{(\boldsymbol{x_3}-\boldsymbol{x_2})^T\boldsymbol{e}}{\|\boldsymbol{e}\|\|\boldsymbol{n_1}\|^2}\boldsymbol{n_1} + \frac{(\boldsymbol{x_4}-\boldsymbol{x_2})^T\boldsymbol{e}}{\|\boldsymbol{e}\|\|\boldsymbol{n_2}\|^2}\boldsymbol{n_2}]$$

$$\frac{\partial}{\partial \boldsymbol{x_2}} w_{b,e}(\theta - \theta^*) = -w_{b,e}[\frac{(\boldsymbol{x_3}-\boldsymbol{x_1})^T\boldsymbol{e}}{\|\boldsymbol{e}\|\|\boldsymbol{n_1}\|^2}\boldsymbol{n_1} + \frac{(\boldsymbol{x_4}-\boldsymbol{x_1})^T\boldsymbol{e}}{\|\boldsymbol{e}\|\|\boldsymbol{n_2}\|^2}\boldsymbol{n_2}]$$

$$\frac{\partial}{\partial \boldsymbol{x_3}} w_{b,e}(\theta - \theta^*) = w_{b,e}\frac{\|\boldsymbol{e}\|}{\|\boldsymbol{n_1}\|^2}\boldsymbol{n_1}$$

$$\frac{\partial}{\partial \boldsymbol{x_4}} w_{b,e}(\theta - \theta^*) = w_{b,e}\frac{\|\boldsymbol{e}\|}{\|\boldsymbol{n_2}\|^2}\boldsymbol{n_2} \tag{27}$$

The derivative of volume constraints with respect to vertex $\boldsymbol{x_k}$ is computed by the sum of the $\boldsymbol{x_k}$'s incident faces, as (28)

$$\frac{\partial}{\partial \boldsymbol{x_k}} w_v(v - v^*) = w_v \sum_{f_{ijk}\in N(\boldsymbol{x_k})} (\boldsymbol{x_i} \times \boldsymbol{x_j}) \tag{28}$$

The derivatives of edge length constraints or dihedral angle constraints with the respect to an interpolation weight α_i is as (29):

$$\frac{\partial}{\partial \alpha_i} w_{s,e}(l - l^*) = -w_{s,e}(L^{(i)} - L)$$

$$\frac{\partial}{\partial \alpha_i} w_{b,e}(\theta - \theta^*) = -w_{b,e}(\Theta^{(i)} - \Theta)$$

$$\frac{\partial}{\partial \alpha_i} w_v(v - v^*) = -w_v(\mathrm{V}^{(i)} - \mathrm{V}) \tag{29}$$

Thirdly, the update direction δ is solved by the equation (30).

$$\boldsymbol{J}(\boldsymbol{x})^T\boldsymbol{J}(\boldsymbol{x})\delta = -\boldsymbol{J}(\boldsymbol{x})^T\boldsymbol{f}(\boldsymbol{x}) \tag{30}$$

In [1], the Jacobian matrix is decomposed in order to take the advantage of the sparse matrix. As mentioned above, except the bottom row, $\boldsymbol{J}(\boldsymbol{x})$ has more zeros than non-zeros. So $\boldsymbol{J}$ is decomposed as (31), where $\boldsymbol{u}$ is the bottom row of the Jacobian matrix and is completely filled; $\tilde{\boldsymbol{J}}$ is the partial derivatives of the edge length constraints and dihedral angles constraints. Let **A** denote the $\boldsymbol{J}^T\boldsymbol{J}$ as (32) and $\boldsymbol{b}$ denote the $-\boldsymbol{J}^T\boldsymbol{f}$ as (34), then equation (30) becomes (34). With the Sherman-Morrison formula in [35], δ is solved by (35).

$$\boldsymbol{J} = \begin{bmatrix} \tilde{\boldsymbol{J}} \\ \boldsymbol{u}^T \end{bmatrix} \tag{31}$$

$$\boldsymbol{J}^T\boldsymbol{J} = \tilde{\boldsymbol{J}}^T\tilde{\boldsymbol{J}} + \boldsymbol{u}\boldsymbol{u}^T = \mathbf{A} + \boldsymbol{u}\boldsymbol{u}^T \tag{32}$$

$$\boldsymbol{b} = -\boldsymbol{J}^T\boldsymbol{f} \tag{33}$$

$$(\mathbf{A} + \boldsymbol{u}\boldsymbol{u}^T)\delta = \boldsymbol{b} \tag{34}$$

$$\delta = (\mathbf{A} + \boldsymbol{u}\boldsymbol{u}^T)^{-1}\boldsymbol{b} = \mathbf{A}^{-1}\boldsymbol{b} - \frac{\mathbf{A}^{-1}\boldsymbol{u}\boldsymbol{u}^T\mathbf{A}^{-1}\boldsymbol{b}}{1+\boldsymbol{u}^T\mathbf{A}^{-1}\boldsymbol{u}} \tag{35}$$

Based on the formulation (36), the solution of δ by equation (30) is computed as follows: ①solving $\boldsymbol{y}$ by $\boldsymbol{Ay} = \boldsymbol{b}$; ②solving $\boldsymbol{z}$ by $\boldsymbol{Az} = \boldsymbol{u}$; ③solving δ by $\delta = \boldsymbol{y} - \frac{\boldsymbol{z}\boldsymbol{u}^T\boldsymbol{y}}{1+\boldsymbol{u}^T\boldsymbol{z}}$. Analyzing the entries of the Jacobian matrix, $\tilde{\boldsymbol{J}}$ is a sparse matrix, so is $\mathbf{A}$. Sparse Cholesky solver is used to solve equation in ① and ②.

Finally, $\boldsymbol{x}$ is updated by δ as (36) until $\mathrm{E}(\boldsymbol{x} + h\delta) < \mathrm{E}(\boldsymbol{x})$ or $h = 10^{-10}$.

$$\boldsymbol{x} = \boldsymbol{x} + h\delta \tag{36}$$

h is the step size starting with 1 and is halved at each iteration. Final position of vertices is obtained.

3.2 Shape Interpolation based on Example-driven Deformation

Generally, shape interpolation has two challenges: vertex correspondence problem and vertex path problem [3]. Vertex correspondence problem is to find the correspondence map between the source model and the target model; vertex path problem is to interpolate the in-between shapes. In the dissertation, it focuses on the interpolation between key-frames. Since two key-frames share the same topology, thus, the correspondence problem is solved. Concerning the vertex path problem, the formulation has two requirements: ①Given the time $t \in [0,1]$, at the time $t = 0$, the shape is required to be the source model (undeformed model); at $t = 1$, the shape turns into the target model (the example); ②the in-between shapes should be natural as much as possible.

Built on the framework of example-driven deformation in Chapter 3.1, the undeformed model is the source model and the example is the target model. At first, vertex coordinates at time t are interpolated linearly. Next the example-driven deformation (18) ~ (21) is employed to optimize the linear-interpolated shape at time t. The shape interpolation is implemented as follows:

Firstly, interpolated coordinates of the in-between shapes are computed by (37).

$$v(t) = (1-t)V + tV_e \qquad t \in [0,1] \tag{37}$$

where v is the vertex coordinates at time t, and V is the corresponding vertex coordinates of the source model, and V_e is the target model's vertex coordinates.

Secondly, the example-driven deformation is employed to optimize the interpolated shape as stated in Chapter 3.1.2.

Chapter4. Experiments

Both the example-driven deformation algorithm and the proposed shape interpolation algorithm is implemented on Windows 10 of the laptop (2.2GHz quad-core Intel Core i7 processor). In this chapter, two parts of experiments are conducted. The first part in Chapter 4.1 is about deformed behavior of Stefan Fröhlich and Mario Botsch's example-driven deformation. The second part in Chapter 4.2 is about the shape interpolation based on the example-driven deformation.

4.1 Example-Driven deformation

This part is experiments to study the deforming behavior of the model by the example-driven deformation in [1]. Two groups are conducted. One is for stretching (Chapter 4.1.1). The other is the rotation (Chapter 4.1.2). Two types of user-interaction are designed: single-selected mode (user selects one vertex and drag it) and multiple-selected mode (user selects several vertices and drag them).

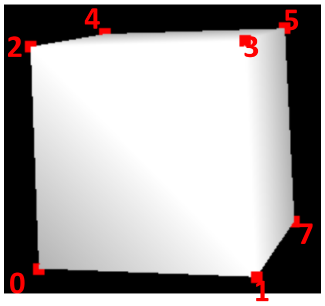
4.1.1 Stretching

Fig.4.1 shows the undeformed box and its example used in the deformation. The undeformed box is shown in Fig.4.1a and the red numbers in the figure are the indices of corresponding vertices. The example is shown in Fig.4.1b. For the group of stretching experiment, two experiments are conducted: the first experiment is to study influence of the stiffness λ and μ; the second experiment is to compare the deformation with single-selected mode and multiple-selected mode.

For the first experiment, the user selects one vertex to input constraints. The selected vertex is vertex 3, noted as blue ball in the following figures. The user moves 1 unit towards the *x*-positive direction (shown in the Before optimization column in the Table 4.1/4.2/4.3/4.4). Also, vertex 0, vertex 1, vertex 6, vertex 7 are set to be unmoved points.

Since the user moves the vertex 3, the coordinates of vertex 3 change more. In addition, vertex 3 and vertex 5 in the example model are positioned at the same level. Therefore, the final deformation struggles to keep them at the same level. As a result, the larger changes of coordinates happen to vertex 3 and vertex 5, as shown in the Table 4.1/4.2/4.3/4.4. The After Optimization column in the table is the results of the example-driven deformation.

Observing results in Table 4.1/4.2/4.3 with different ratio of λ to μ, the deformation with same ratio of λ and μ (λ/μ) is almost the same and the tiny difference is negligible. Table 4.1 and Fig.4.2 show the results of $\lambda/\mu = 1$; Table 4.2 and Fig.4.3 show the results of $\lambda/\mu = 10$; Table 4.3 shows the results of $\lambda/\mu = 100$; and Table 4.4 shows the results of $\lambda/\mu = 1000$ and $\lambda/\mu = 10000$. Looking at all results, $\lambda/\mu = 1, 10{,}100$ have better deformation compared to $\lambda/\mu = 1000{,}10000$, since x coordinate of vertex 3 is much closer to the user's constraints and, also, the vertex 3 and vertex 5 are better kept at the same level.



a. Undeformed box

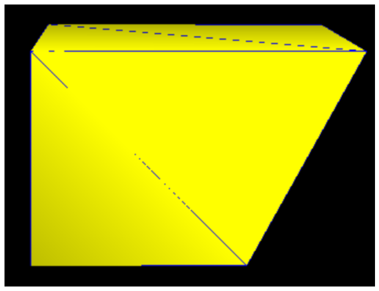
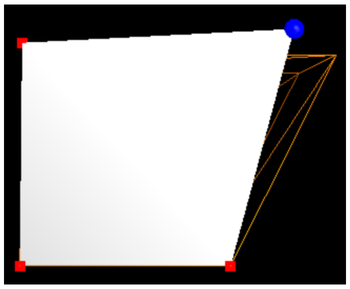

b. Example

Fig.4.1 The undeformed box and its example model


a. Deformed box's front view

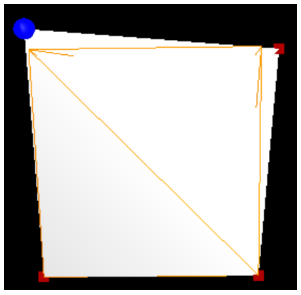
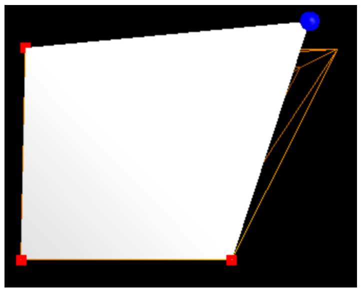

b. Deformed box's side view

Fig.4.2 The deformed box of $\lambda/\mu = 1$


a. Deformed box's front view

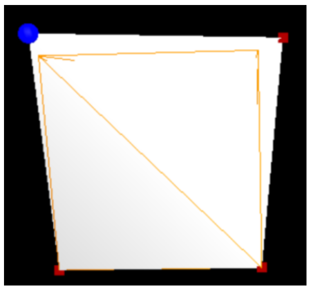
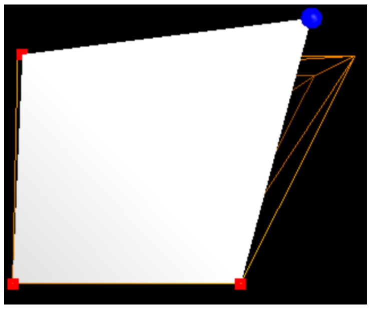

b. Deformed box's side view

Fig.4.3 The deformed box of $\lambda/\mu = 10$


a. Deformed box's front view

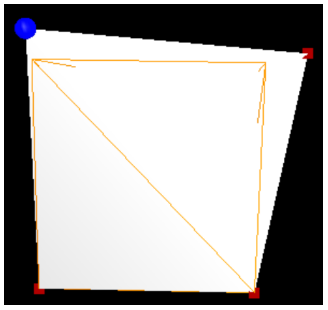
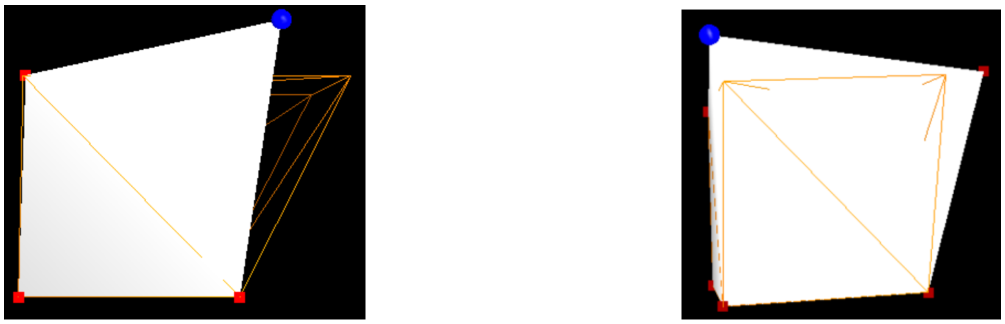

b. Deformed box's side view

Fig.4.4 The deformed box of $\lambda/\mu = 100$


a. The deformed box of $\lambda/\mu = 1000$

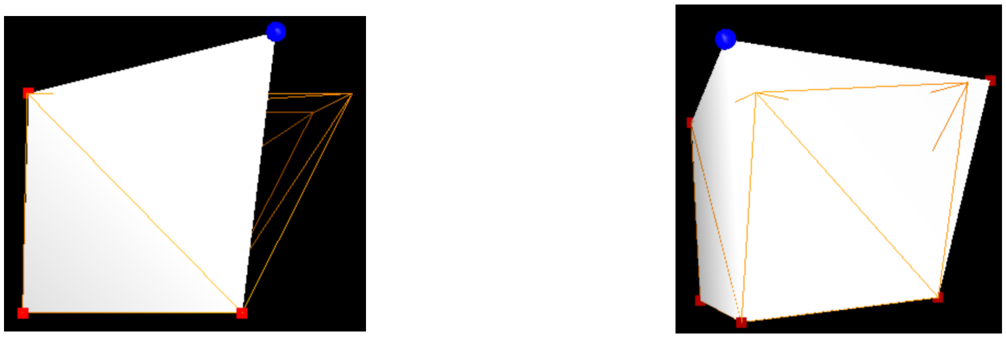

b. The deformed box of $\lambda/\mu = 10000$

Fig.4.5 The deformed box of $\lambda/\mu = 1000$ and $\lambda/\mu = 10000$

Table 4.1 Deformation with $\lambda/\mu = 1$

Parameters	Vertex index	Before optimization	After optimization
λ=100, μ=100, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.508428 0.556346 0.53441]
	3	[1.5 0.5 0.5]	[0.831318 0.622191 0.557532]
	4	[-0.5 0.5 -0.5]	[-0.551219 0.58137 -0.557703]
	5	[0.5 0.5 -0.5]	[0.735219 0.527227 -0.609216]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	209.789	30.3803
λ=1000, μ=1000, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.508089 0.554075 0.532691]
	3	[1.5 0.5 0.5]	[0.83025 0.618887 0.557355]
	4	[-0.5 0.5 -0.5]	[-0.549771 0.578654 -0.557148]
	5	[0.5 0.5 -0.5]	[0.735672 0.527234 -0.608125]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	2634.72	109.523
λ=10000, μ=10000, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.506186 0.541335 0.52305]
	3	[1.5 0.5 0.5]	[0.824255 0.600351 0.556363]
	4	[-0.5 0.5 -0.5]	[-0.541642 0.563417 -0.554032]
	5	[0.5 0.5 -0.5]	[0.738217 0.527272 -0.602008]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	25847.2	718.316

Table 4.2 Deformation with $\lambda/\mu = 10$

Parameters	Vertex index	Before optimization	After optimization
λ=100, μ=10, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.498215 0.505204 0.511749]
	3	[1.5 0.5 0.5]	[0.897911 0.628247 0.581973]
	4	[-0.5 0.5 -0.5]	[-0.565476 0.519324 -0.507354]
	5	[0.5 0.5 -0.5]	[0.779837 0.592922 -0.632333]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	159.259	18.5842
λ=1000, μ=100, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.497921 0.505201 0.511424]
	3	[1.5 0.5 0.5]	[0.897393 0.628048 0.582023]
	4	[-0.5 0.5 -0.5]	[-0.565037 0.519125 -0.50745]
	5	[0.5 0.5 -0.5]	[0.779286 0.592665 -0.632074]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	1092.59	60.8
λ=10000, μ=1000, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.495482 0.505177 0.50873]
	3	[1.5 0.5 0.5]	[0.893098 0.626394 0.582434]
	4	[-0.5 0.5 -0.5]	[-0.561396 0.517476 -0.508239]
	5	[0.5 0.5 -0.5]	[0.774711 0.590539 -0.629924]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	10425.9	475.983

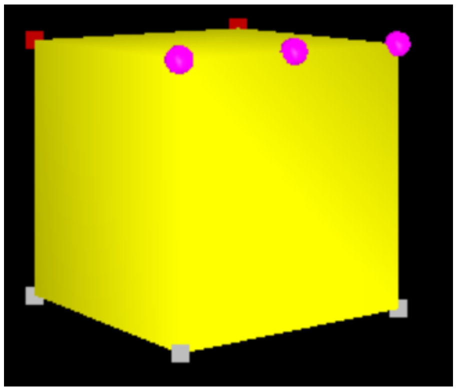
Table 4.3 Deformation with $\lambda/\mu = 100$

Parameters	Vertex index	Before optimization	After optimization
λ=100, μ=1, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.476954 0.506314 0.507733]
	3	[1.5 0.5 0.5]	[0.840337 0.665908 0.563986]
	4	[-0.5 0.5 -0.5]	[-0.537636 0.503367 -0.490774]
	5	[0.5 0.5 -0.5]	[0.75251 0.582275 -0.725946]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	143.838	24.983
λ=1000, μ=10, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.47697 0.506311 0.507742]
	3	[1.5 0.5 0.5]	[0.840385 0.665891 0.563978]
	4	[-0.5 0.5 -0.5]	[-0.537652 0.50337 -0.490768]
	5	[0.5 0.5 -0.5]	[0.752565 0.582292 -0.725964]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	938.379	81.9701
λ=10000, μ=100, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.47712 0.50628 0.507829]
	3	[1.5 0.5 0.5]	[0.84085 0.665731 0.563896]
	4	[-0.5 0.5 -0.5]	[-0.537799 0.5034 -0.490714]
	5	[0.5 0.5 -0.5]	[0.753087 0.582452 -0.726142]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	8883.79	651.673

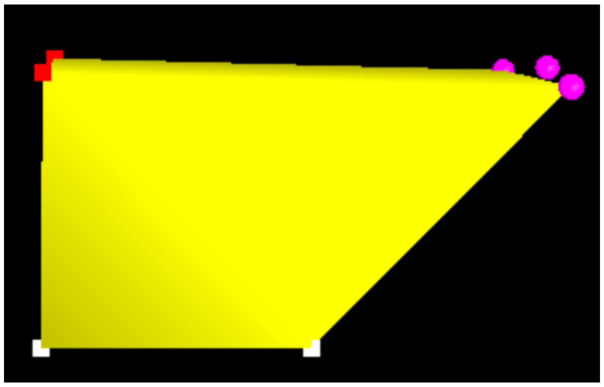
Table 4.4 Deformation with $\lambda/\mu = 1000$ and $\lambda/\mu = 10000$

Parameters	Vertex index	Before optimization	After optimization
λ=1000, μ=1, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.487584 0.502918 0.499175]
	3	[1.5 0.5 0.5]	[0.720252 0.761746 0.539686]
	4	[-0.5 0.5 -0.5]	[-0.530355 0.499643 -0.49973]
	5	[0.5 0.5 -0.5]	[0.670854 0.553758 -0.817446]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	922.958	140.993
λ=10000, μ=1, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.493328 0.501254 0.496723]
	3	[1.5 0.5 0.5]	[0.685556 0.790617 0.534509]
	4	[-0.5 0.5 -0.5]	[-0.531431 0.499165 -0.502892]
	5	[0.5 0.5 -0.5]	[0.646267 0.546303 -0.839044]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	8714.16	1325.68

In the second experiment, the undeformed box and the example are the same as the first experiment. Vertex 0, vertex 1, vertex 6, vertex 7 are also set to be unmoved points. The only difference is that two vertices are selected. Vertex 3 and vertex 5 are selected, since they are the ones moved in the example. As shown in the Fig.4.6, purple dots are two selected vertices and the middle purple dot is the center of vertex 3 and vertex 5. The input constraints are applied for vertex 3 and vertex 5, as shown in the Before optimization column in the Table 4.5. The deformation results are shown in the After optimization column (Fig.4.6b). Deformation of different λ/μ is shown in the Fig.4.7. Looking at all results, $\lambda/\mu = 1{,}10{,}100$ have the better deformation, as concluded in the first experiment. Also, deformation with two-selected constraints are better than the single-selected. As shown in the Table 4.5, vertex's coordinates are much closer to the user's inputs and, obviously, vertex 3 and vertex 5 are at the same level. The deformation with another example is shown in the Fig.4.8. Fig.4.8a is the example; Fig.4.8b is the deformation with single-selected vertex; Fig.4.8c is the deformation with two-selected vertices.

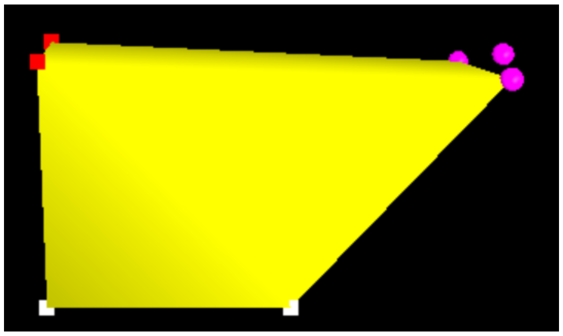


a. Undeformed box of selected vertices

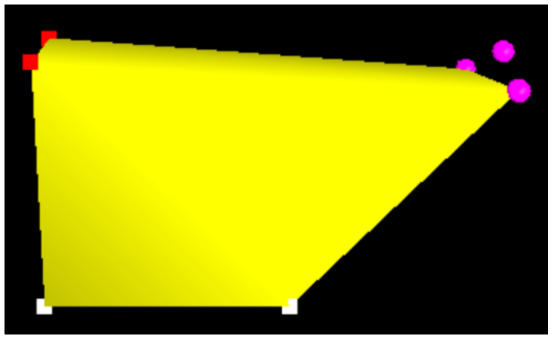


b. Deformed box of $\lambda/\mu = 1$

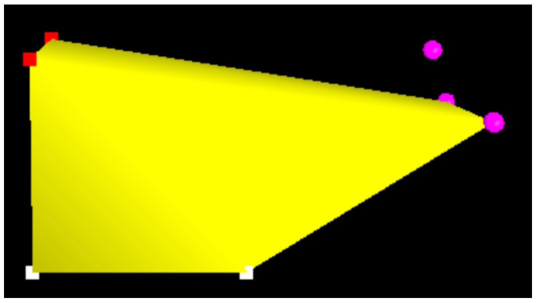
Fig.4.6 Undeformed box and deformed box of $\lambda/\mu = 1$



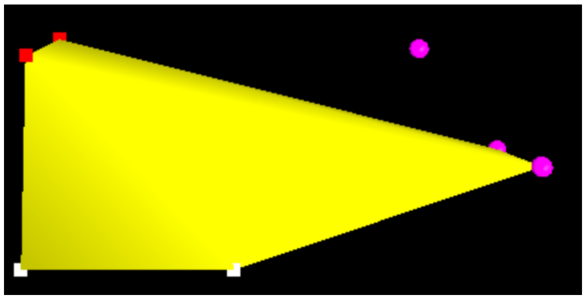
a. Deformed box of $\lambda/\mu = 10$



b. Deformed box of $\lambda/\mu = 100$

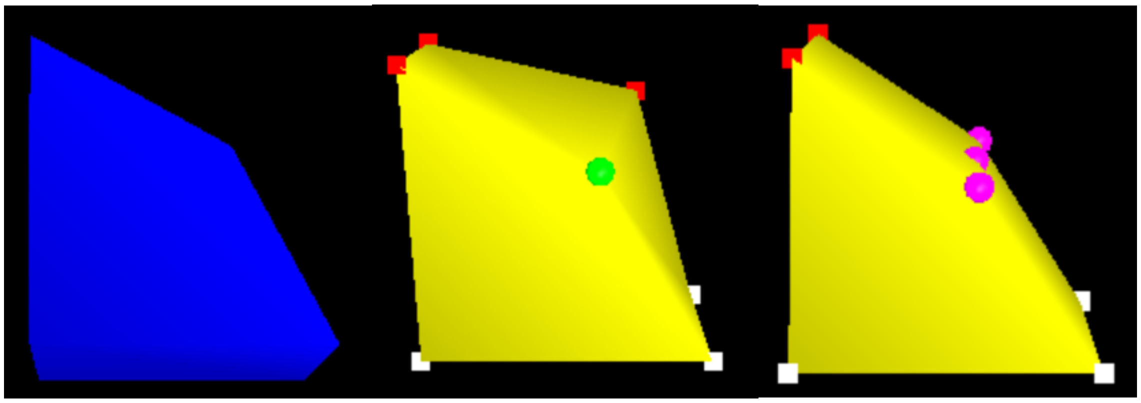


c. Deformed box of $\lambda/\mu = 1000$



d. Deformed box of $\lambda/\mu = 10000$

Fig.4.7 Deformed box of different λ/μ



a. Example b. Single-selected vertex c. Two-selected vertices

Fig. 4.8 Deformation with single selected vertex and two-selected vertices

Table 4.5 Deformation with two selected vertices

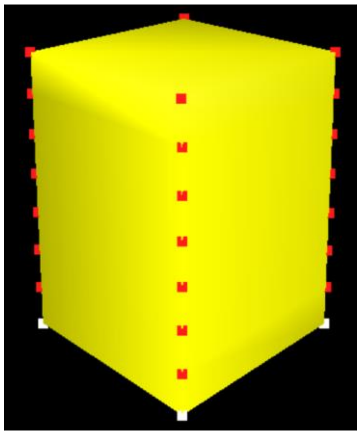
Parameters	Vertex index	Before optimization	After optimization
λ=100, μ=100, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.497567 0.522288 0.503185]
	3	[1.5 0.5 0.5]	[1.46429 0.470712 0.497883]
	4	[-0.5 0.5 -0.5]	[-0.534649 0.49681 -0.523258]
	5	[1.5 0.5 -0.5]	[1.44036 0.444141 -0.535191]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	309.298	3.04618
λ=100, μ=10, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.540117 0.510268 0.524525]
	3	[1.5 0.5 0.5]	[1.41856 0.436746 0.47262]
	4	[-0.5 0.5 -0.5]	[-0.570738 0.515664 -0.510966]
	5	[1.5 0.5 -0.5]	[1.41816 0.420113 -0.550793]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	281.539	0.903098
	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]

λ=100, μ=1, υ=1000	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.551157 0.501417 0.547113]
	3	[1.5 0.5 0.5]	[1.45553 0.374851 0.434253]
	4	[-0.5 0.5 -0.5]	[-0.565728 0.525205 -0.475104]
	5	[1.5 0.5 -0.5]	[1.46279 0.365962 -0.585179]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	278.763	0.542208
λ=1000, μ=1, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.511356 0.505673 0.546926]
	3	[1.5 0.5 0.5]	[1.68117 0.194636 0.435698]
	4	[-0.5 0.5 -0.5]	[-0.483892 0.52128 -0.467947]
	5	[1.5 0.5 -0.5]	[1.72105 0.163394 -0.57973]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	1659.86	14.8973
λ=10000, μ=1, υ=1000	0	[-0.5 -0.5 0.5]	[-0.5 -0.5 0.5]
	1	[0.5 -0.5 0.5]	[0.5 -0.5 0.5]
	2	[-0.5 0.5 0.5]	[-0.477064 0.509602 0.520008]
	3	[1.5 0.5 0.5]	[1.96519 -0.0198982 0.468254]
	4	[-0.5 0.5 -0.5]	[-0.375663 0.511912 -0.489349]
	5	[1.5 0.5 -0.5]	[2.07323 -0.107296 -0.54129]
	6	[-0.5 -0.5 -0.5]	[-0.5 -0.5 -0.5]
	7	[0.5 -0.5 -0.5]	[0.5 -0.5 -0.5]
	Energy	15470.8	2.62972

4.1.2 Rotation

Fig.4.9 shows the undeformed model and its example used in the deformation. The undeformed model is shown in Fig.4.9a and the example is shown in Fig.4.9b. Similar to experiments in the stretching, two experiments are conducted in this part. However, as shown in Fig.4.10, the single-selected deformation is unnatural. The green dot is the single-selected vertex. With dragging this vertex, all other vertices act like the example. Fig.4.10a shows the deformation with the green dot moving by a small step. Fig.4.10b shows the result of the green dot by a large step. Because of the unnatural deformation of single-selected, the multi-selected mode is employed only. The stiffness' influence is studied in the following.

During the experiment, the four vertices at the bottom of the model are set to be the unmoved points, as white dots shown in the figure. Fig.4.11 shows the deformation of different λ/μ after dragging the 4 vertices (purple dots) along the replacement vector [0.8,0.4,0]; Fig.4.11a,b,c,d show the deformed model of $\lambda/\mu = 10{,}100{,}1000{,}10000$ respectively. Fig.4.12 shows the deformation of different λ/μ after dragging them along the replacement vector continuously. Comparing the deformed model in Fig.4.11 and Fig.4.12, $\lambda/\mu = 100{,}1000$ have more natural results. Table 4.6~Table 4.9 (Appendix A) shows the coordinates of the deformed model in Fig.4.11.


a. Undeformed model

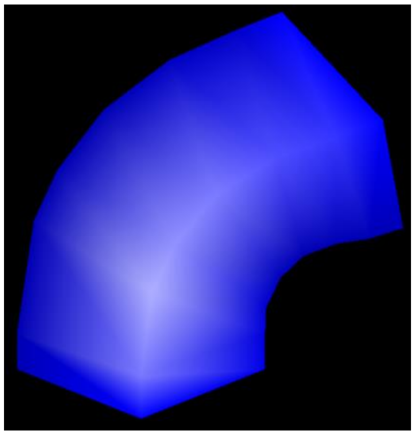
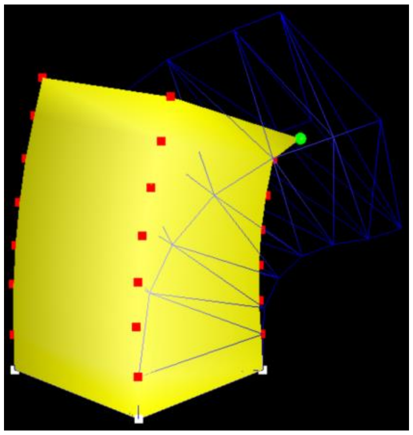

b. Example

Fig.4.9 The undeformed model and its example


a. Deformed model with small movement

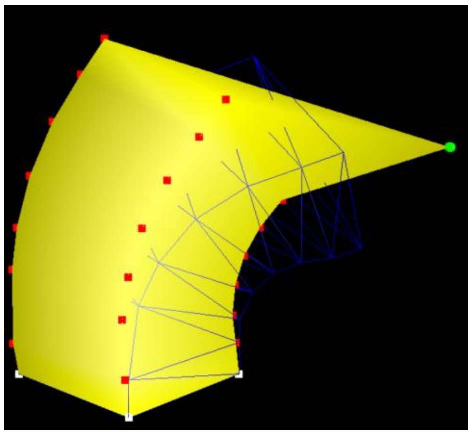
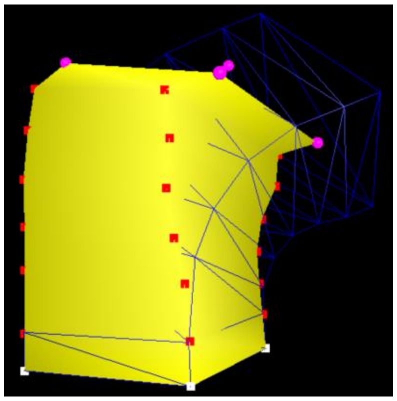
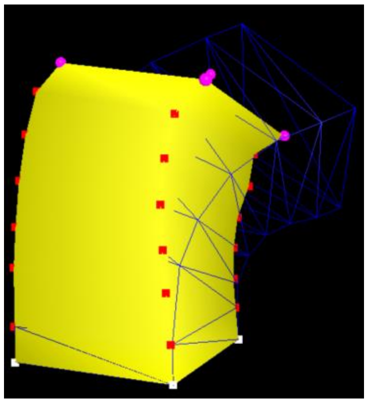
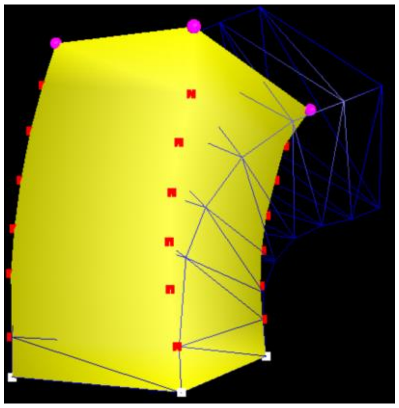

b. Deformed model with large movement

Fig.4.10 The undeformed model and its example


a. Deformed model of $\lambda/\mu = 10$


b. Deformed model of $\lambda/\mu = 100$


c. Deformed model of $\lambda/\mu = 1000$

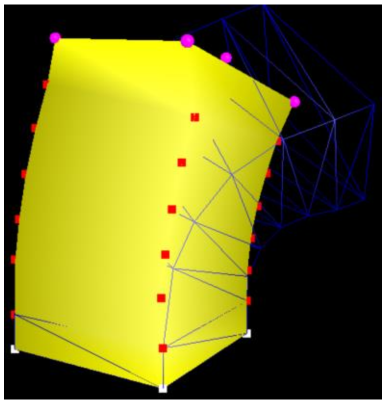
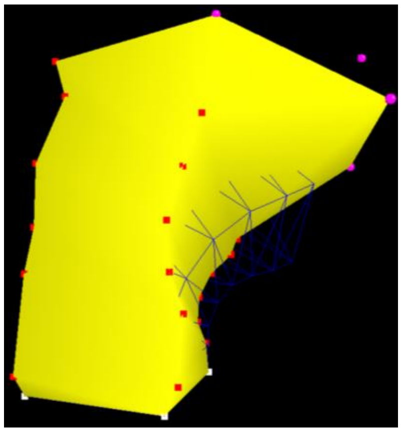
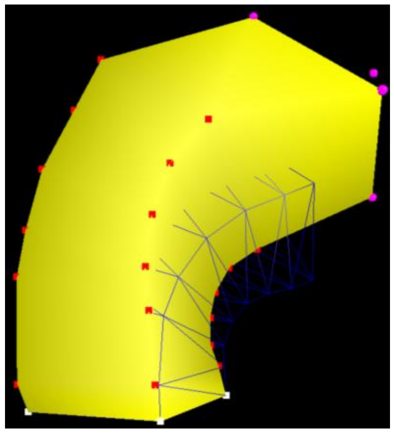

d. Deformed model of $\lambda/\mu = 10000$

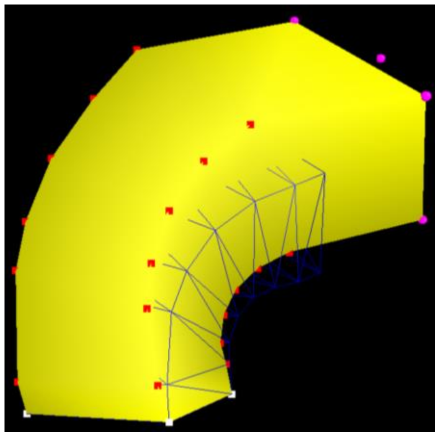
Fig.4.11 The Deformation of different λ/μ



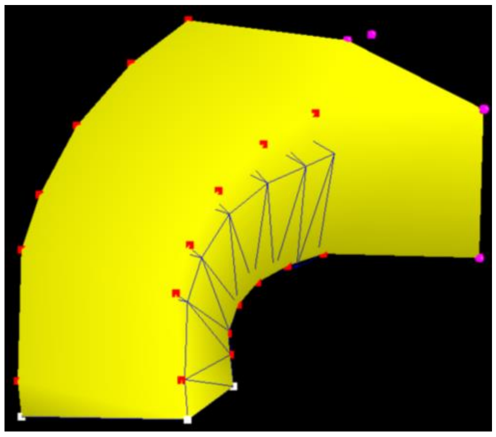
a. Deformed model of $\lambda/\mu = 10$



b. Deformed model of $\lambda/\mu = 100$



c. Deformed model of $\lambda/\mu = 1000$



d. Deformed model of $\lambda/\mu = 10000$

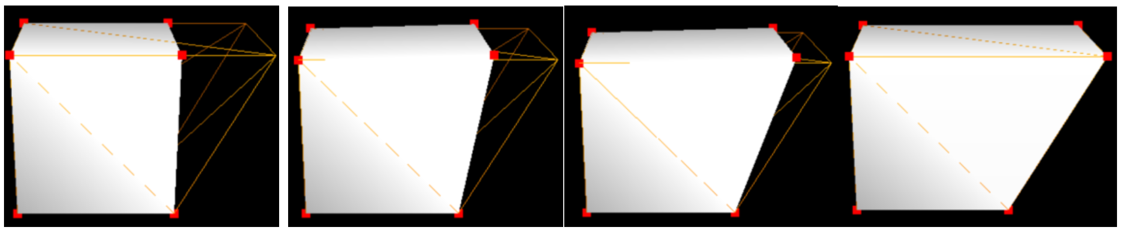
Fig.4.12 The Deformation of different λ/μ after dragging several times

4.2 Shape Interpolation based on Example-Driven Deformation

As mentioned in Chapter 3.2, the in-between shapes are interpolated based on the example-driven deformation. The experiment is to study the performance of stretching and rotation with different λ/μ. The time t is set to 0.35 and 0.65.

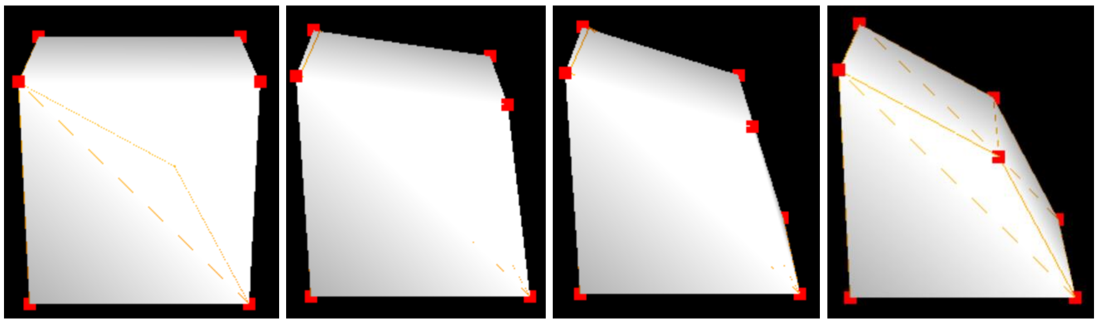
Fig.4.13 and Fig.4.14 show the stretching performance of the shape interpolation of example-driven deformation. The same box is stretching towards different target model through $t = 0$, $t = 0.35$, $t = 0.65$ and $t = 1$ with $\lambda/\mu = 100$. The orange frame is the shape of the target model. Since different λ/μ has very little effect on the interpolated shapes, here, just the $\lambda/\mu = 100$ is presented. As stretching is linear compared to nonlinear rotation, it is easy to obtain the natural in-between shapes even by linear interpolation only, while the rotation interpolation is the challenge with linear interpolation. Traditional shape interpolation is to interpolate the shape in translation and rotation separately. With Stefan Fröhlich and Mario Botsch's example-driven deformation, it is unnecessary to interpolate separately. The experiments of rotation are conducted with different angles. Fig.4.15 shows the process of cylinder rotating 60° with $\lambda/\mu = 100$. Different λ/μ have little effect on its results. Nevertheless, except the interpolation of 60° rotation, experiments for 120° rotation and 180° rotation have more natural in-between shapes with λ/μ=10000; and other ratio of λ to μ produces unnatural shapes. Fig.4.16 shows the process of rotating 120°. Fig.4.17 and Fig.4.18 show the process of rotating 180° with different cylinder.

In conclusion, this shape interpolation has natural shape for rotation within 180°. Experiments for angles larger than 180° have the unnatural shapes.



a. *t*=0 b. *t*=0.35 c. *t*=0.65 d. *t*=1

Fig.4.13 The box stretching process with $\lambda/\mu = 100$



a. *t*=0 b. *t*=0.35 c.*t*=0.65 d.*t*=1

Fig.4.14 The box stretching process with $\lambda/\mu = 100$

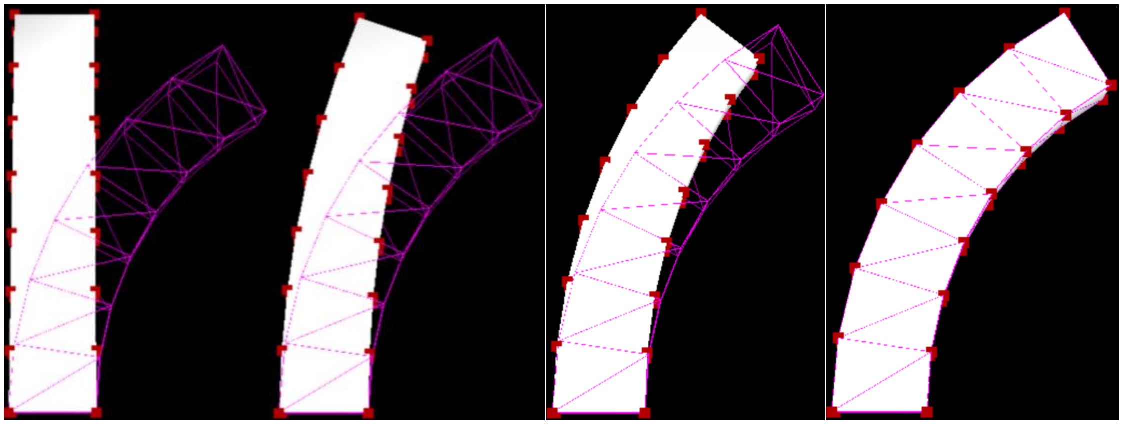
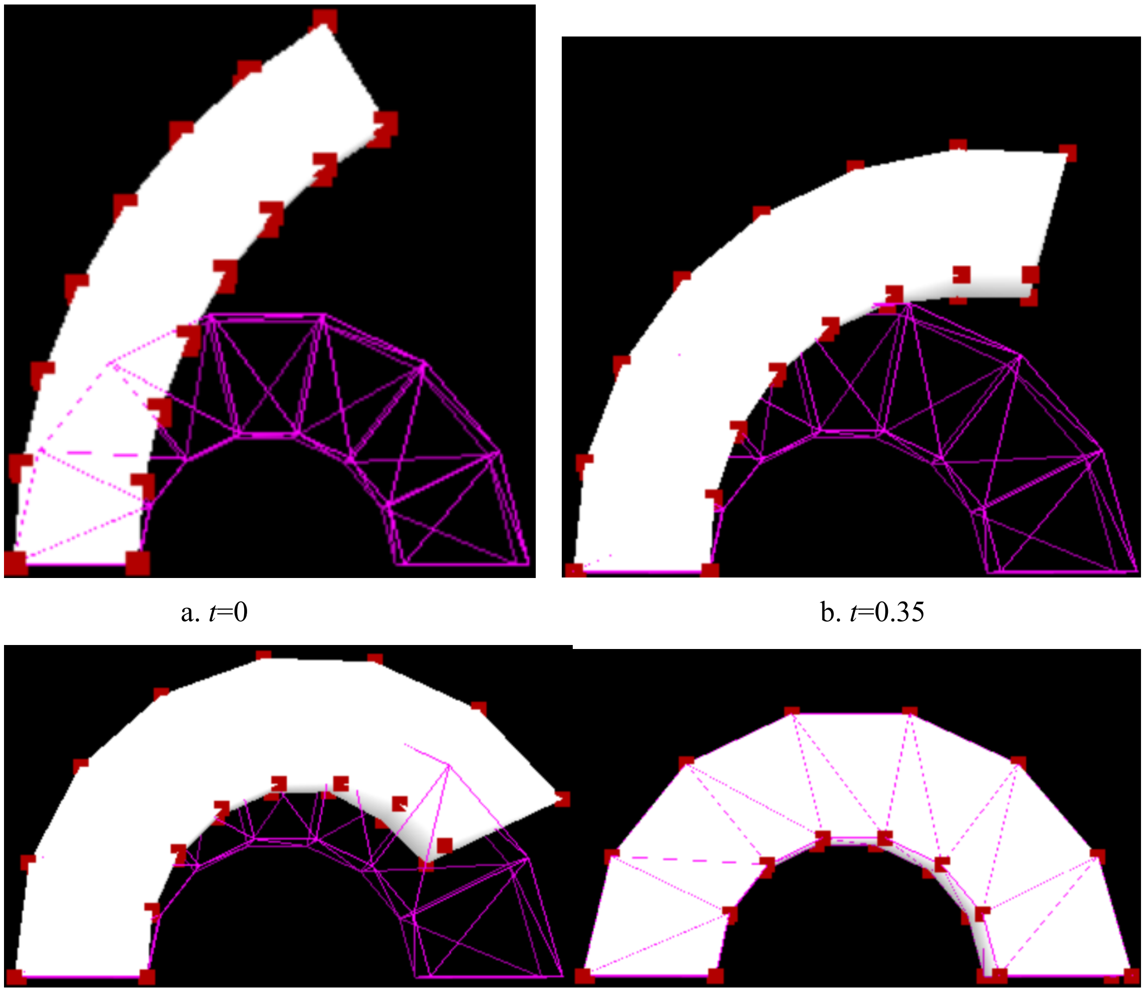
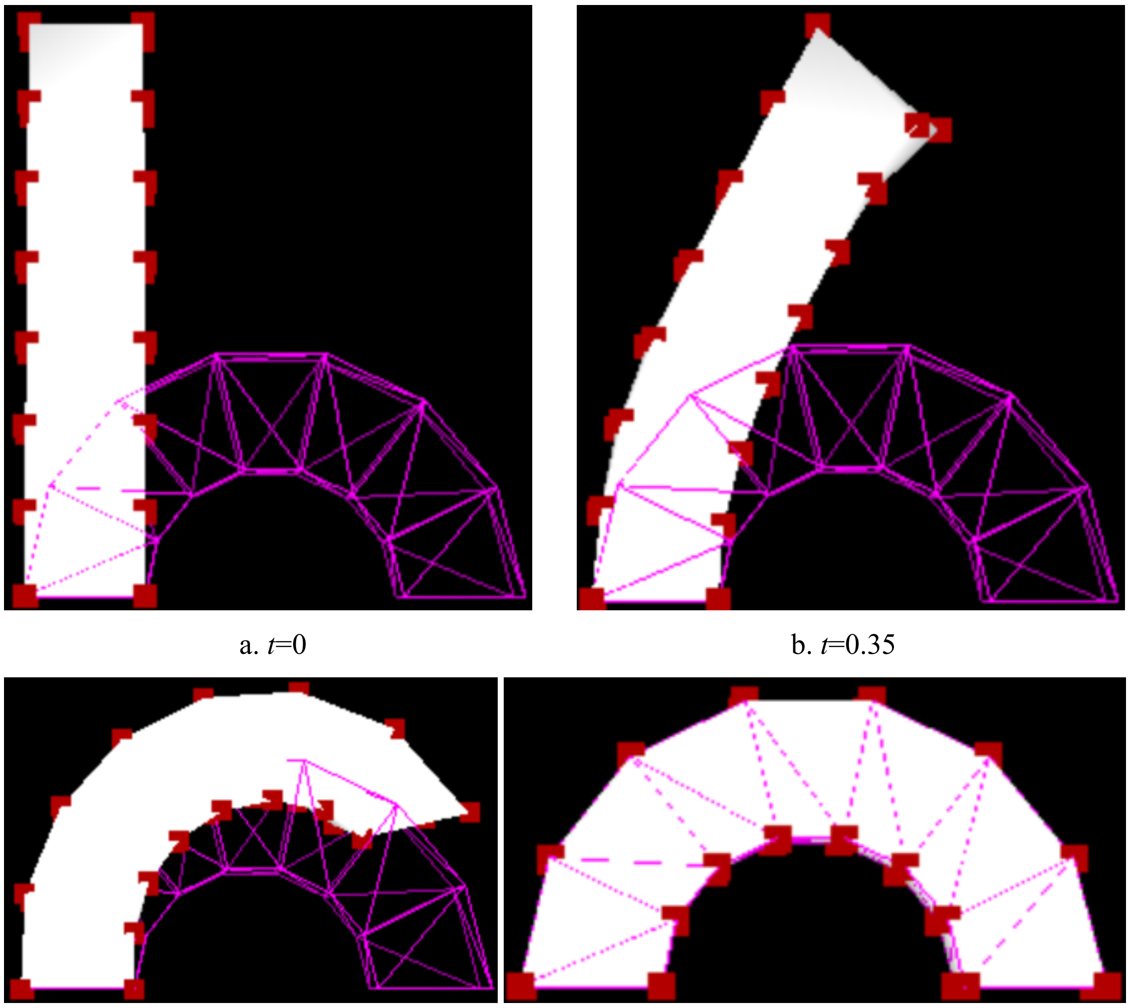

a. t=0 b. t=0.35 c.t=0.65 d.t=1

Fig.4.15 The 60° rotation of a straight cylinder with $\lambda/\mu = 100$


a. t=0 b. t=0.35

c.t=0.65 d.t=1

Fig.4.16 The 120° rotation of a 60°-cylinder with $\lambda/\mu = 10000$

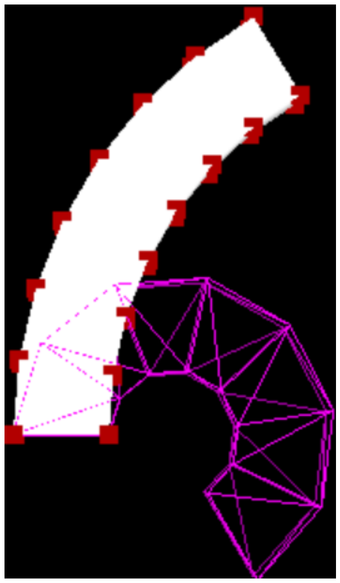

a. t=0 b. t=0.35

c.t=0.65 d.t=1

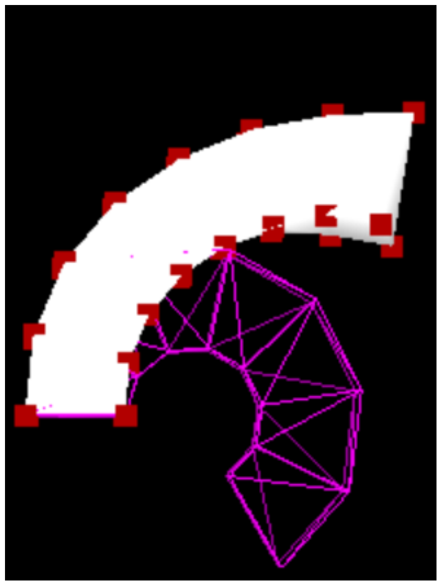
Fig.4.17 The 180°rotation of a straight cylinder with $\lambda/\mu = 10000$

4.3 Discussion

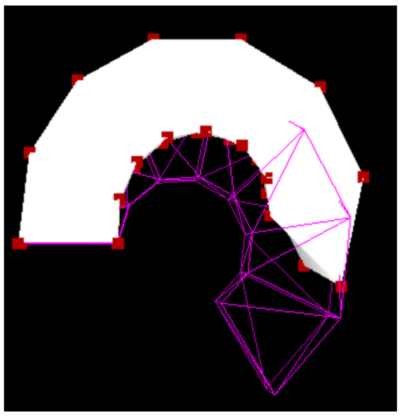
From the experiments of example-driven deformation, it is concluded that multiple-selected vertices have better performance and low ratio of λ to μ does better in the stretching deformation and high ratio of λ to μ is better at rotation. A similar conclusion in the shape interpolation experiments is that higher ratio of λ to μ has better performance at rotation. As λ is the stiffness of stretching and μ is the one of bending, lower ratio of λ to μ means that objects are easier to stretch relative to high ratio; higher ratio of λ to μ represents objects easier to bend relative to low ratio.



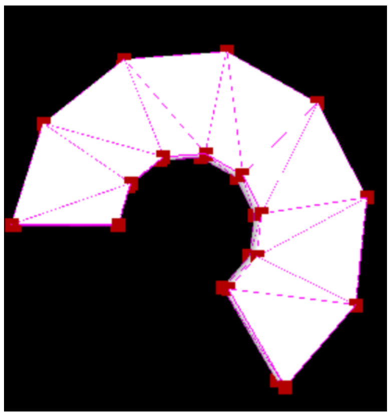
a. t=0



b. t=0.35



c.t=0.65



d.t=1

Fig.4.18 The 240°rotation of a 60°-cylinder with $\lambda/\mu = 10000$

Chapter5. Conclusion

Different from traditional shape interpolation approaches interpolating the translation and rotation separately, the method applied in the dissertation interpolates translation and rotation in the meantime. It is built on the framework of example-driven deformation proposed by Stefan Fröhlich and Mario Botsch, which combines the physics-based deformation with the example-driven deformation. In the dissertation, I study and implement the Stefan Fröhlich and Mario Botsch's example-driven deformation. A series of experiments have been conducted to demonstrate the good performance of this deformation. Then, the shape interpolation built on this example-driven deformation is stated and implemented. Experiments demonstrate the good performance of the interpolation in both stretching and less-than-180° rotation.

References

[1] **Fröhlich** S, Botsch M. Example‐Driven Deformations Based on Discrete Shells[C]//Computer graphics forum. Blackwell Publishing Ltd, 2011, 30(8): 2246-2257.

[2] **Sederberg** T W, Parry S R. Free-form deformation of solid geometric models[J]. ACM SIGGRAPH computer graphics, 1986, 20(4): 151-160.

[3] **Jung** Y, Graf H, Behr J, et al. Mesh deformations in X3D via CUDA with freeform deformation lattices[M]//Virtual and Mixed Reality-Systems and Applications. Springer Berlin Heidelberg, 2011: 343-351.

[4] **Griessmair** J, Purgathofer W. Deformation of solids with trivariate B-splines[C]//Proceedings of eurographics. 1989, 89: 137-148.

[5] **Feng** J, Heng P A, Wong T T. Accurate B-spline free-form deformation of polygonal objects[J]. Journal of Graphics Tools, 1998, 3(3): 11-27.

[6] **Coquillart** S. Extended free-form deformation: a sculpturing tool for 3D geometric modeling[M]. ACM, 1990.

[7] http://web.cs.wpi.edu/~matt/courses/cs563/talks/smartin/ffderform.html

[8] **Singh** K, Fiume E. Wires: a geometric deformation technique[C]//Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM, 1998: 405-414.

[9] **Lounsbery** M, DeRose T D, Warren J. Multiresolution analysis for surfaces of arbitrary topological type[J]. ACM Transactions on Graphics (TOG), 1997, 16(1): 34-73.

[10] **Botsch** M, Sorkine O. On linear variational surface deformation methods[J]. IEEE transactions on visualization and computer graphics, 2008, 14(1): 213-230.

[11] **Kobbelt** L P, Vorsatz J, Labsik U. A shrink wrapping approach to remeshing polygonal surfaces[C]//Computer Graphics Forum. Blackwell Publishers Ltd, 1999, 18(3): 119-130.

[12] **Kobbelt** L, Campagna S, Vorsatz J, et al. Interactive multi-resolution modeling on arbitrary meshes[C]//Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM, 1998: 105-114.

[13] **Kobbelt** L P, Bareuther T, Seidel H P. Multiresolution shape deformations for

meshes with dynamic vertex connectivity[C]//Computer Graphics Forum. Blackwell Publishers Ltd, 2000, 19(3): 249-260.

[14] **Sorkine** O, Cohen-Or D, Lipman Y, et al. Laplacian surface editing[C]//Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. ACM, 2004: 175-184.

[15] **Xu** W W, Zhou K. Gradient domain mesh deformation—a survey[J]. Journal of computer science and technology, 2009, 24(1): 6-18.

[16] **Yu** Y, Zhou K, Xu D, et al. Mesh editing with poisson-based gradient field manipulation[C]//ACM Transactions on Graphics (TOG). ACM, 2004, 23(3): 644-651.

[17] **Zayer** R, Rössl C, Karni Z, et al. Harmonic guidance for surface deformation[C]//Computer Graphics Forum. Blackwell Publishing, Inc, 2005, 24(3): 601-609.

[18] **Huang** J, Shi X, Liu X, et al. Subspace gradient domain mesh deformation[C]//ACM Transactions on Graphics (TOG). ACM, 2006, 25(3): 1126-1134.

[19] **Steihaug** T. An inexact gauss-newton approach to mildly nonlinear problems[R]. Tech. rep., Dept. of Mathematics, University of Linkoping, 1995.

[20] **Au** O K C, Tai C L, Liu L, et al. Dual Laplacian editing for meshes[J]. IEEE Transactions on Visualization and Computer Graphics, 2006, 12(3): 386-395.

[21] **Lipman** Y, Sorkine O, Cohen-Or D, et al. Differential coordinates for interactive mesh editing[C]//Shape Modeling Applications, 2004. Proceedings. IEEE, 2004: 181-190.

[22] **Celniker** G, Gossard D. Deformable curve and surface finite-elements for free-form shape design[J]. ACM SIGGRAPH computer graphics, 1991, 25(4): 257-266.

[23] **Terzopoulos** D, Platt J, Barr A, et al. Elastically deformable models[C]//ACM Siggraph Computer Graphics. ACM, 1987, 21(4): 205-214.

[24] **Botsch** M, Sorkine O. On linear variational surface deformation methods[J]. IEEE transactions on visualization and computer graphics, 2008, 14(1): 213-230.

[25] **Welch** W, Witkin A. Variational surface modeling[C]//ACM SIGGRAPH computer graphics. ACM, 1992, 26(2): 157-166.

[26] **Sumner** R W, Zwicker M, Gotsman C, et al. Mesh-based inverse kinematics[J].

ACM transactions on graphics (TOG), 2005, 24(3): 488-495.

[27] **Alexa** M, Cohen-Or D, Levin D. As-rigid-as-possible shape interpolation[C]//Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 2000: 157-164.

[28] **Xu D**, Zhang H, Wang Q, et al. Poisson shape interpolation[J]. Graphical Models, 2006, 68(3): 268-281.

[29] **Kilian M**, Mitra N J, Pottmann H. Geometric modeling in shape space[C]//ACM Transactions on Graphics (TOG). ACM, 2007, 26(3): 64.

[30] **Chao I**, Pinkall U, Sanan P, et al. A simple geometric model for elastic deformations[J]. ACM Transactions on Graphics (TOG), 2010, 29(4): 38.

[31] **Sederberg** T W, Greenwood E. A physically based approach to 2–D shape blending[C]//ACM SIGGRAPH computer graphics. ACM, 1992, 26(2): 25-34.

[32] **Winkler T**, Drieseberg J, Alexa M, et al. Multi‐Scale Geometry Interpolation[C]//Computer graphics forum. Blackwell Publishing Ltd, 2010, 29(2): 309-318.

[33] **Grinspun** E, Hirani A N, Desbrun M, et al. Discrete shells[C]//Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association, 2003: 62-67.

[34] **Bridson** R, Marino S, Fedkiw R. Simulation of clothing with folds and wrinkles[C]//Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation. Eurographics Association, 2003: 28-36.

[35] **Golub** G H, Van Loan C F. Matrix computations[M]. JHU Press, 2012.

Appendix A

Table 4.6 Deformation with $\lambda/\mu = 10$

	Vertex index	Before optimization	After optimization
λ=100, μ=10, υ=1000	0	[0 -1 -1]	[0 -1 -1]
	1	[-1 -1 -0]	[-1 -1 -0]
	2	[-0 -1 1]	[-0 -1 1]
	3	[1 -1 0]	[1 -1 0]
	4	[0 -0.714286 -1]	[-0.00236351 -0.699982 -1.01276]
	5	[-1 -0.714286 -0]	[-0.995302 -0.725892 -0.00163193]
	6	[-0 -0.714286 1]	[0.00717483 -0.709508 0.997646]
	7	[1 -0.714286 0]	[0.983475 -0.723871 0.00980615]
	8	[0 -0.428571 -1]	[0.0113705 -0.309259 -1.02333]
	9	[-1 -0.428571 -0]	[-0.987906 -0.246736 -0.0141555]
	10	[-0 -0.428571 1]	[-0.00726858 -0.328474 0.986404]
	11	[1 -0.428571 0]	[0.978938 -0.471533 -0.00689171]
	12	[0 -0.142857 -1]	[0.0309478 0.01504 -1.00499]
	13	[-1 -0.142857 -0]	[-0.978482 0.0805652 -0.0373812]
	14	[-0 -0.142857 1]	[-0.0641252 -0.0144275 0.963226]
	15	[1 -0.142857 0]	[0.956328 -0.201718 0.0291153]
	16	[0 0.142857 -1]	[0.0745705 0.363593 -0.998935]
	17	[-1 0.142857 -0]	[-0.979357 0.460383 -0.0617219]
	18	[-0 0.142857 1]	[-0.106837 0.324249 0.980825]
	19	[1 0.142857 0]	[0.97328 0.0641911 0.125538]
	20	[0 0.428571 -1]	[0.196243 0.721287 -1.03085]
	21	[-1 0.428571 -0]	[-0.932921 0.85684 -0.104068]
	22	[-0 0.428571 1]	[-0.0815981 0.673338 1.01236]
	23	[1 0.428571 0]	[1.07769 0.334005 0.216969]
	24	[0 0.714286 -1]	[0.322598 1.06882 -1.01102]
	25	[-1 0.714286 -0]	[-0.856963 1.22739 -0.244515]
	26	[-0 0.714286 1]	[-0.0986401 1.0113 1.00352]
	27	[1 0.714286 0]	[1.17748 0.606967 0.0911514]

	28	[0.8 1.4 -1]	[0.462345 1.39007 -0.923711]
	29	[-0.2 1.4 0]	[-0.607512 1.42326 -0.11696]
	30	[0.8 1.4 1]	[0.242381 1.10013 1.20262]
	31	[1.8 1.4 0]	[1.46843 0.693897 0.254992]
	Energy	1556.14	159.656

Table 4.7 Deformation with $\lambda/\mu = 100$

	Vertex index	Before optimization	After optimization
λ=100, μ=1, υ=1000	0	[0 -1 -1]	[0 -1 -1]
	1	[-1 -1 -0]	[-1 -1 -0]
	2	[-0 -1 1]	[-0 -1 1]
	3	[1 -1 0]	[1 -1 0]
	4	[0 -0.714286 -1]	[0.00634842 -0.711901 -0.999507]
	5	[-1 -0.714286 -0]	[-0.991407 -0.716255 -0.00685212]
	6	[-0 -0.714286 1]	[-0.0032295 -0.712874 1.0059]
	7	[1 -0.714286 0]	[0.986899 -0.716073 0.0129492]
	8	[0 -0.428571 -1]	[0.0476412 -0.288712 -1.0159]
	9	[-1 -0.428571 -0]	[-0.950424 -0.12433 -0.0315967]
	10	[-0 -0.428571 1]	[-0.00419944 -0.288439 1.00949]
	11	[1 -0.428571 0]	[0.984599 -0.468178 0.0282208]
	12	[0 -0.142857 -1]	[0.136935 0.0263669 -1.01895]
	13	[-1 -0.142857 -0]	[-0.865701 0.231018 -0.056525]
	14	[-0 -0.142857 1]	[0.0282913 0.0330777 1.01173]
	15	[1 -0.142857 0]	[1.02694 -0.19628 0.0616944]
	16	[0 0.142857 -1]	[0.295857 0.359351 -1.02966]
	17	[-1 0.142857 -0]	[-0.721675 0.650434 -0.0973415]
	18	[-0 0.142857 1]	[0.101978 0.385246 1.02007]
	19	[1 0.142857 0]	[1.10666 0.0638102 0.103475]
	20	[0 0.428571 -1]	[0.514761 0.679851 -1.05827]
	21	[-1 0.428571 -0]	[-0.519175 1.07311 -0.138485]
	22	[-0 0.428571 1]	[0.231145 0.73686 1.03925]
	23	[1 0.428571 0]	[1.25546 0.310361 0.1627]

	24	[0 0.714286 -1]	[0.750956 0.971727 -1.04816]
	25	[-1 0.714286 -0]	[-0.292888 1.44054 -0.17148]
	26	[-0 0.714286 1]	[0.397057 1.05624 1.07374]
	27	[1 0.714286 0]	[1.43414 0.550102 0.129406]
	28	[0.8 1.4 -1]	[1.03336 1.12064 -0.952698]
	29	[-0.2 1.4 0]	[-0.0508261 1.68394 -0.164401]
	30	[0.8 1.4 1]	[0.573613 1.34804 1.18405]
	31	[1.8 1.4 0]	[1.69916 0.666338 0.341032]
	Energy	1670.56	16.236

Table 4.8 Deformation with $\lambda/\mu = 1000$

	Vertex index	Before optimization	After optimization
λ=1000, μ=1, υ=1000	0	[0 -1 -1]	[0 -1 -1]
	1	[-1 -1 -0]	[-1 -1 -0]
	2	[-0 -1 1]	[-0 -1 1]
	3	[1 -1 0]	[1 -1 0]
	4	[0 -0.714286 -1]	[0.0107854 -0.714004 -0.99771]
	5	[-1 -0.714286 -0]	[-0.992261 -0.714493 -0.00988647]
	6	[-0 -0.714286 1]	[-0.00906522 -0.714363 1.00416]
	7	[1 -0.714286 0]	[0.989313 -0.714765 0.0165174]
	8	[0 -0.428571 -1]	[0.057684 -0.294467 -1.01273]
	9	[-1 -0.428571 -0]	[-0.953077 -0.123454 -0.0438178]
	10	[-0 -0.428571 1]	[-0.015136 -0.292448 1.00293]
	11	[1 -0.428571 0]	[0.990322 -0.46524 0.0341563]
	12	[0 -0.142857 -1]	[0.145396 0.0183486 -1.01351]
	13	[-1 -0.142857 -0]	[-0.872428 0.233664 -0.0600532]
	14	[-0 -0.142857 1]	[0.0306396 0.0255039 1.01165]
	15	[1 -0.142857 0]	[1.04446 -0.193096 0.0585079]
	16	[0 0.142857 -1]	[0.292476 0.351205 -1.01548]
	17	[-1 0.142857 -0]	[-0.729773 0.653977 -0.0812421]
	18	[-0 0.142857 1]	[0.126301 0.370224 1.01921]
	19	[1 0.142857 0]	[1.14299 0.065232 0.0842081]

	20	[0 0.428571 -1]	[0.493496 0.674954 -1.02035]
	21	[-1 0.428571 -0]	[-0.527743 1.07736 -0.098224]
	22	[-0 0.428571 1]	[0.279136 0.710868 1.02873]
	23	[1 0.428571 0]	[1.29623 0.312341 0.111269]
	24	[0 0.714286 -1]	[0.713054 0.970269 -1.00933]
	25	[-1 0.714286 -0]	[-0.308517 1.44945 -0.0990629]
	26	[-0 0.714286 1]	[0.466852 1.01886 1.04576]
	27	[1 0.714286 0]	[1.47521 0.553595 0.116888]
	28	[0.8 1.4 -1]	[1.05505 1.03097 -0.900837]
	29	[-0.2 1.4 0]	[-0.114254 1.75984 -0.182511]
	30	[0.8 1.4 1]	[0.500738 1.50054 1.14126]
	31	[1.8 1.4 0]	[1.67715 0.755389 0.410995]
	Energy	16392.8	411.239

Table 4.9 Deformation with $\lambda/\mu = 10000$

	Vertex index	Before optimization	After optimization
λ=1000, μ=1, υ=1000	0	[0 -1 -1]	[0 -1 -1]
	1	[-1 -1 -0]	[-1 -1 -0]
	2	[-0 -1 1]	[-0 -1 1]
	3	[1 -1 0]	[1 -1 0]
	4	[0 -0.714286 -1]	[0.0042954 -0.714227 -0.99802]
	5	[-1 -0.714286 -0]	[-0.997677 -0.714309 -0.00251083]
	6	[-0 -0.714286 1]	[-0.00226204 -0.714346 1.00152]
	7	[1 -0.714286 0]	[0.997674 -0.714451 0.00537492]
	8	[0 -0.428571 -1]	[0.0384045 -0.303009 -1.01025]
	9	[-1 -0.428571 -0]	[-0.965962 -0.142294 -0.027194]
	10	[-0 -0.428571 1]	[0.00273936 -0.301744 0.992962]
	11	[1 -0.428571 0]	[1.00494 -0.462314 0.00915745]
	12	[0 -0.142857 -1]	[0.107333 0.00967217 -1.00968]
	13	[-1 -0.142857 -0]	[-0.895979 0.211492 -0.0312965]
	14	[-0 -0.142857 1]	[0.0598236 0.0123916 0.995495]

	15	[1 -0.142857 0]	[1.06125 -0.18942 0.0161675]
	16	[0 0.142857 -1]	[0.227058 0.343952 -1.01099]
	17	[-1 0.142857 -0]	[-0.768672 0.625836 -0.0391641]
	18	[-0 0.142857 1]	[0.16385 0.350311 0.99544]
	19	[1 0.142857 0]	[1.15802 0.07056 0.0223655]
	20	[0 0.428571 -1]	[0.398091 0.672152 -1.01382]
	21	[-1 0.428571 -0]	[-0.58378 1.04402 -0.0475553]
	22	[-0 0.428571 1]	[0.31861 0.684247 0.994661]
	23	[1 0.428571 0]	[1.30075 0.321427 0.0279096]
	24	[0 0.714286 -1]	[0.589144 0.973779 -1.01275]
	25	[-1 0.714286 -0]	[-0.380169 1.41375 -0.0508783]
	26	[-0 0.714286 1]	[0.497507 0.99077 0.996996]
	27	[1 0.714286 0]	[1.46588 0.567532 0.0310763]
	28	[0.8 1.4 -1]	[0.912423 1.056 -0.97177]
	29	[-0.2 1.4 0]	[-0.225666 1.77289 -0.219384]
	30	[0.8 1.4 1]	[0.460674 1.56558 1.0404]
	31	[1.8 1.4 0]	[1.60254 0.858862 0.285602]
	Energy	163615	6548.05