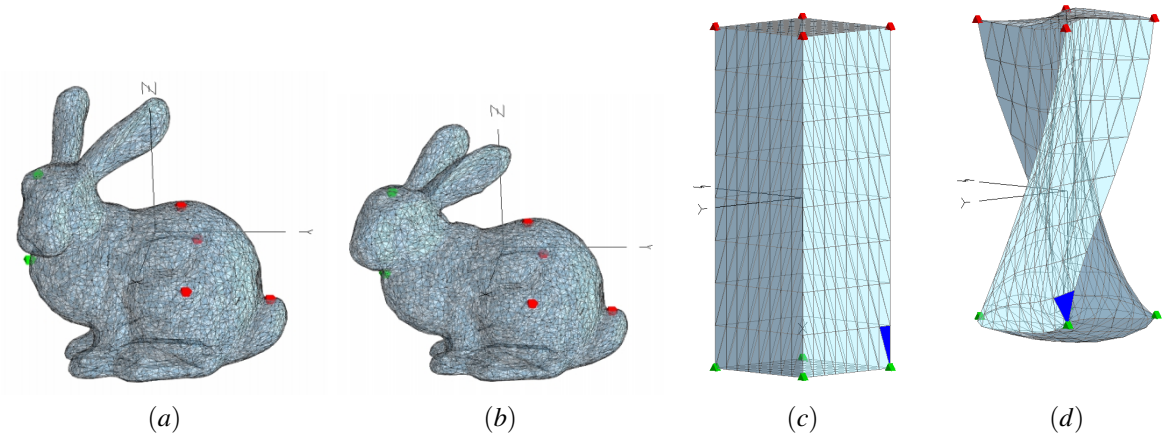
Realization of Linear Rotation-invariant Coordinates for Meshes

Tianle Yuan*
Oregon State University



(a) (b) (c) (d)

Figure 1: (a) and (b) are the deformed Bunny model by implementing the rotation-invariant mesh edition with constrains of 6 anchor points (2 soft constrain points which are marked by the green dots and 4 hard constrain points which are marked by the red dots). For the deformation effect, we translates the 2 soft constrain points in the negative direction of Z axis and rotate them around the negative direction of Z axis. (c) and (d) are the screwed Cuboid model by rotating $\frac{\pi}{2}$ around the X axis.

Abstract

We implements a rigid motion invariance for the discrete 3D mesh deformation based on the method introduced in *Linear Rotation-invariant Coordinates for Meshes* [Lipman et al. 2005]. To realize this kind of deformation, we introduce the discrete frame for the mesh representation. The geometry reconstruction can be separated into two linear sub steps: the construction of local frame for the mesh and the Laplacian deformation for the new mesh frame. The two sub steps are related to two linear sparse equation systems according to their energy formulas. To satisfy the requirement of nonsingular matrix system, the surface editing operations also include the local frames' orientation constrain and the vertices positions' constrain from users. In this paper, we shows the realization of the basic deformation effects based on mesh transformations.

Keywords: mesh representation, discrete frame, deformation, sparse linear system.

1 Introduction

This paper mainly do the implementation of the rigid motion invariant mesh representation technique. With the representation focusing on a local region of the discrete surface, the reconstruction of the mesh geometry can nicely preserve the local surface appearance when some part of the mesh is edited by user. The local property preservation only located between neighbour vertices. When the locations of the mesh's vertices change, then vertices orientation properties would not be changed but keep good directional transfer. The technique provide a direction for solving the morbid effect problem of the Laplacian surface deformation.

The representation traverse to all the vertices on the mesh. For each vertex, they have their own local discrete frame. The rotation interpolation between two adjacent vertices is based on their relative coordinates transition. In this paper, we create our discrete frame for each vertex based on the vertex basis introduced by the paper named *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds* [Meyer et al. 2003]. The discrete frame based on the surface normal and tangential plane can be more helpful for the further curvature visualization.

There are two main steps in the deformation pipeline. One is the reconstructions for the local frames. By expressing the adjacent matrix for mesh's vertices discrete frame and combining the orientation constrains, all vertices' frame will be rebuilt. The other step is the reconstructions for the mesh position. In our implementation, we combined Laplacian deformation [Sorkine et al. 2004] in the mesh editing step. We tried to use the local discrete frames interactions to eliminate the inflexible orientation changes of the local Laplacian vector. The deformed results come from the solution from the two large sparse linear systems by using the least-square method.

The main implementation in this paper can be shown as:

1. We represent discrete frame for each vertex and the local discrete frame can be updated by given orientation constrains.
2. We correct Laplacian vector's directions by referring each vertex's local frame.
3. We realize some simple interactive mesh edition when keep-

*e-mail: {yuant}@eecs.oregonstate.edu

ing the local properties.

2 Previous Work

There has been a number of work in the development of interactive mesh editing [Kobbelt et al. 1998; Botsch and Kobbelt 2004; Yu et al. 2004; Sorkine et al. 2004]. With the maturity of 3D printing technology, the visual modification of geometric mesh data has become particularly important. It is the main optimization content of mesh editing tool to modify the geometry shape while maintaining the original geometry surface details.

Multiresolution Approches: By applying fast mesh smoothing algorithm and mesh decimation algorithms, the simplified meshes will generate a process hierarchy [Kobbelt et al. 1998]. The interactive mesh modification is down in the local frame in the low resolution mesh. The editing constrain the supports regions and piece-wise boundary conditions when supporting the smoothness ranging continuously from C_0 to C_2 [Botsch and Kobbelt 2004]. By reconstructing the edited mesh, the final deform result can be done according to the hierarchy. The mesh edited result almost looks the same as the Poisson and Laplacian methods.

Poisson Gradient Mesh Editing: The method modifies the mesh through the manipulation on the gradient field [Yu et al. 2004]. The modification which is based on the Possion equation, is implicit. When the triangle meshes are teared by the local frame transformation, the Poisson equation stitches the individual triangles into a new polygon mesh. However, the orientation of all the local frames need to be manipulated explicitly. That is to say, user should define the transformations for all the vertices of handles.

Laplacian Mesh Editing: This is a method focusing on the preservation of the surface differencial properties [Sorkine et al. 2004]. With the user-defined positional constrain of mesh vertices, the Laplacian operator can do implicit mesh transformations for whole the geometry. However, the energy function of Laplacian method keeps the surface Laplacian vector always in the same directions. It cause the method is sensitive to vertices rotation and scale transformations. Thus, like the Poisson-based mesh editing, the method cannot modify the local frames correctly to adapt the user-defined deformation constrains.

Harmonic Field: In [Zayer et al. 2005], it provides harmonic fields for the preservation of mesh's local properties and global geometry. By applying the harmonic fields, the transformation constrain on user-defined handles is interpolated into the whole mesh model. With the transformation interpolation, the further Laplacian editing preserves more of the original local features.

Rotation-Invariant Pyramid Coordinates: In [Sheffer and Kraevoy 2004], the method consider the local quantities into surface representation. The rotattion-invariant pyramid coordinates, which constructed by the tangent plane and normal of discrete surface, have been introduced for the mesh representation with any transformation. For this method, it based on nonlinear equation system.

3 Background

3.1 3D Mesh and Mesh Representation

3D model is the product after 3D modeling or 3D scanning. The name *3D mesh* in this paper refers to 3D shell or 3D boundary,which is one of the two catagories of 3D model [Wikipedia contributors 2021a]. To eliminate the topology inconvenience, such as planarity, convexity, and constancy of face and edges,of concave surface, 3D triangular mesh can fast implement discrete Boolean logic, surface smoothing, and surface simplification.

For 3D triangular mesh, there are several common representations [Wikipedia contributors 2021d]:

Face-Vertex Meshes: No matter loading .ply file or .obj file, the basic data are the position of geometry's vertices locations and triangular topology for mesh triangle connecting. The triangle structure saves the list of vertices. The vertex structure records its global coordinates in the 3D space and saves the index of all its 1-neighbour vertices and triangular faces. The face-vertex representation is the basic mesh presenting format.

Winged edges: Based on the relationship between vertices and faces, winged edges describes new relationships both between edge and faces, and between edge and vertices. Each winged edge structure saves two faces and two vertices. By adding this structure, structures of faces and vertices will also record the corresponding indices of edges.

Polyhedron: The structure polyhedron is a collection of all the exist mesh representation structures. Structures like vertex, faces, and edges, all recorded in the polyhedron structure in the form of index lists. It is a smallest unit of a mesh primitive. Once there are some manipulations, such as mesh smoothing, subdivision, simplification, for the geometry mesh, the polyhedron is the minimal manipulating unit. This means all the parameters of representation members of the polyhedron structure such as the coordinates of vertices, the connecting topology of faces, the number of edges, need to be traversed and updated.

Corner tables: To emphasize relationship between vertices and their nearest neighbor vertices, edges,and faces, corner table provides a convenient way for fast geometry elements indexing. Corner tables can not represent geometry meshes completely [Wikipedia contributors 2021d]. There are six kind of indexing. They are the vertex of the corner, the edge of the corner,the triangle of the corner, the next corner, the previous corner and the opposite corner. With those retrieving, the surface expressed by triangle fans can be used to represent the discrete case of a curved surface.

With the mature expression of 3D triangular mesh, in the field of mesh construction such as 3D printing, manipulating the model meshes as the user wanted became an imperative task.

3.2 Laplace Operator and Laplacian Mesh Editing

The Laplace operator is an operator express the differential property of a surface. The Laplace operator operating on a function equals getting the divergence of the gradient of a function on Euclidean space [Wikipedia contributors 2021b]. The Laplacian of surface function f can be defined as (1) showing. In (1), u_i is the coordinate basis of the Euclidean space.

$$\begin{aligned} \Delta f = \nabla^2 f &= \nabla \cdot \nabla f \\ &= \sum_{i=1}^{n} \frac{\partial^2 f}{\partial u_i^2} \end{aligned} \tag{1}$$

In differential geometry, for functions defined on submanifolds (or manifold triangle mesh) in Euclidean space, the Laplace–Beltrami operator is a generalization of the Laplace operator [Wikipedia contributors 2021c]. For function f at a vertex V_i, its Laplacian can be approximated as (2).

$$\Delta f(V_i) \approx \frac{1}{2\mathscr{A}^i_{Mixed}} \sum_{j \in N_1(i)} (cot\ \alpha_{ij} + cot\ \beta_{ij})\,(V_i - V_j) \tag{2}$$

Where $\mathscr{A}^i_{Mixed}$ is the Voronoi region of the V_i under the normal case, the sum is over all 1-ring adjacent vertices V_j of V_i, and α_{ij} and

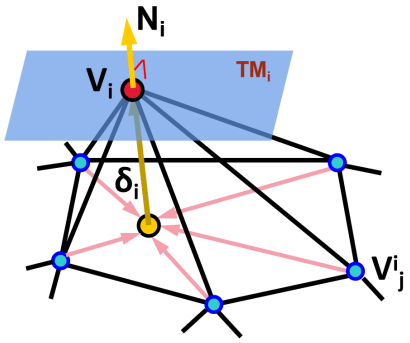


Figure 2: The Laplacian vector on a discrete surface. V_i is the center vertex, V_j^i is the 1-ring neighbour of the center vertex. $\boldsymbol{\delta_i}$ is the Laplacian coordinate (Laplacian vector) of the center vertex. $\mathbf{TM_i}$ is denoted as the tangent plane go through the center vertex. $\mathbf{N_i}$ is the normal of the center vertex.

β_{ij} are the two angles opposing the edge E_j^i. In (2), the cotangent scheme can guarantee the Laplacian vector δ_i (Laplacian coordinates) is in the same direction of normal of the tangent plane TM_i of V_i, which can be seen in 2. The weight scheme can also be the uniform scheme, the cord scheme, and the tangent scheme.

With the expression of Laplace–Beltrami operator, we define the operator with the uniform scheme as $\mathscr{L}$. Then we can construct the Laplacian coordinates at V_i on mainfold triangle mesh as in 3.

$$\delta_i = \mathscr{L}(V_i) = \sum_{j \in N_1(i)} \omega_{ij} \, (V_i - V_j) = V_i - \sum_{j \in N_1(i)} \omega_{ij} \, V_j \tag{3}$$

The equation can be also expressed as the matrix format for the polyheron mesh that need to be deformed:

$$\mathbf{LV} = \delta \tag{4}$$

From Equation (4), $\mathbf{L}$ is the coeffieient matrix of the Laplacian operation, which is the same in Equation (4). $\mathbf{V}$ is the coordinates of the mesh vertices, δ is the vector of Laplacian coorinates.

Obviously, to deform the mesh of geometry, we need to make sure that the linear equation system of Laplacian coordinates exist unique solution.

Thus, additional conditions need to set the left side matrix of the Equation (4) with full rank. Then, we add positional constrains as the initial conditions. Those positional constrains comes from the points that user defines. We call them anchor points. There are two classes of anchor points according to their functions. The first class is the soft constrain points (SCP). By setting SCP, the additional conditions records the translate vector. The second class is the hard constrain points (HCP). By setting HCP, the additional conditions lock the position of the point, which also means the translate vector of the vertex is 0.

Hence, the Laplacian-based mesh editing (deformation) can be concluded as a over-constrained optimization problem. The formula can be represented as a minimized error expression:

$$\begin{cases} E(\mathbf{V}') = \sum\limits_{i=1}^{n} \| \delta_i - \mathscr{L}(V_i') \|^2 + \sum\limits_{i=m}^{n} \| V_i' - U_i \|^2 \\ \mathbf{V}' = \underset{\mathbf{V}'}{arg\,min}(E(\mathbf{V}')) \end{cases} \tag{5}$$

.

The Equation (5) can be expressed as a matrix format:

$$\begin{pmatrix} \mathbf{L} \\ \mathbf{I_{rank=k}} \end{pmatrix} \mathbf{V}' = \begin{pmatrix} \delta \\ \mathbf{h} \end{pmatrix} \tag{6}$$

From (6), $\mathbf{L}$ is the coeffieient matrix of the Laplacian operation, which is the same in (4). $\mathbf{V}'$ is the new coordinates of the mesh vertices, δ is the vector of Laplacian coorinates, which can be calculated from (3). $\mathbf{I_{rank=k}}$ is a irregular unit matrix whose rank is k, which is larger than the number of the lack rank of $\mathbf{L}$. $\mathbf{h}$ is the vector of updated coordinates of anchor points. The detail parameter of the equation can be seen in Equation (7). $\omega_{\mathbf{nj_p^n}}$ is denoted as the weight of the pth 1-ring neighbour of the vertex V_n.

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & -\omega_{0 j_1^0} & \cdots & 0 \\ 0 & 1 & \cdots & \cdots & -\omega_{1 j_k^1} & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -\omega_{n j_p^n} & -\omega_{n j_{p+1}^n} & \cdots & 1 \\ \hline 0 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & 1 & 0 \end{array}\right] \left[\begin{array}{c} V_0' \\ V_1' \\ \vdots \\ V_n' \end{array}\right] = \left[\begin{array}{c} \delta_0 \\ \delta_1 \\ \vdots \\ \delta_n \\ \hline \\ U_2 \\ U_p \\ \vdots \\ U_{n-1} \end{array}\right] \tag{7}$$

To get the Laplasian mesh editing result, the solutions of (6) or (7) are the answer. The matrix equation system can be classified as linear sparse equation system with over-constrained conditions. Thus it is efficient to get the answer if we use the least-square method after doing Cholesky decomposition.

The Laplacian coordinates have three significant properties, which can be used for explainting the morbid character of the Laplacian mesh editing:

(a) The first one is the linear transformation. Assuming T is the linear transformation on the vertices collection V of the geometry, then:

$$T(\mathscr{L}(V_i)) \equiv \mathscr{L}(T(V_i)) \tag{8}$$

(b) The second one is the translation invariance. Assuming Assuming T is the translation on the geometry vertices, then:

$$\begin{aligned} \sum_{j \in N_1(i)} \omega_{ij} \, (TV_i - TV_j) &= \sum_{j \in N_1(i)} \omega_{ij} \, T(V_i - V_j) \\ &= \sum_{j \in N_1(i)} \omega_{ij} T \begin{pmatrix} x_{v_i} - x_{v_j} \\ y_{v_i} - y_{v_j} \\ z_{v_i} - z_{v_j} \\ 0 \end{pmatrix} \\ &= \sum_{j \in N_1(i)} \omega_{ij} \begin{pmatrix} x_{v_i} - x_{v_j} \\ y_{v_i} - y_{v_j} \\ z_{v_i} - z_{v_j} \\ 0 \end{pmatrix} \\ &= \sum_{j \in N_1(i)} \omega_{ij} \, (V_i - V_j) = \delta_i \end{aligned} \tag{9}$$

(c) The third one is the ssensitivity of rotation transformation. Assuming R is the rotation on the geometry vertices, then:

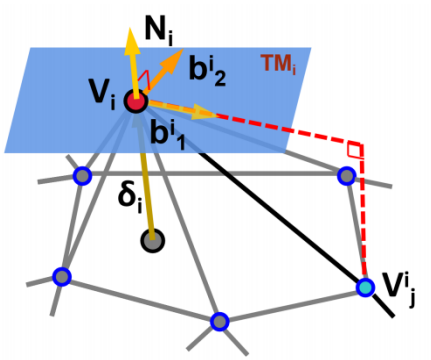


Figure 3: The discrete frame of the vertex V_i. $\mathbf{N_i}$ is the normal of the center vertex V_i. $\mathbf{b^i_1}$ and $\mathbf{b^i_2}$ are the orthogonal tangent vectors on the tangent plane $\mathbf{TM_i}$.

$$\sum_{j\in N_1(i)} \omega_{ij}\,(RV_i - RV_j) = R\sum_{j\in N_1(i)} \omega_{ij}\,(V_i - V_j) = R\delta_i \qquad (10)$$

From 8, 9, 10, we can see that Laplacian coordinates are good at mesh translation but sensitive to vertices rotate and scale transformation. Thus, when users try to rotate and scale the anchor points, the extra factor will stretch the Laplacian coordinates.

4 Technique

4.1 Discrete Frame

From [Lipman et al. 2005], to represent the existing geometry mesh, it introduces a new concept named discrete frame. As Figure 3 shows, for each vertex V_i, the triplet $(\mathbf{b^i_1}, \mathbf{b^i_2}, \mathbf{N_i})$ is the discrete frame of the vertex. Thus, the variable quantity of the discrete frame of each vertex can be expressed as:

$$\Delta(\mathbf{F_i}) = \begin{pmatrix} \delta_j(\mathbf{b^i_1}) \\ \delta_j(\mathbf{b^i_2}) \\ \delta_j(\mathbf{N^i}) \end{pmatrix} = \begin{pmatrix} \mathbf{b^j_1} - \mathbf{b^i_1} \\ \mathbf{b^j_2} - \mathbf{b^i_2} \\ \mathbf{N^j} - \mathbf{N^i} \end{pmatrix} \qquad (11)$$

Thus the discrete surface can be represented as:

$$\begin{aligned} &\Delta(\mathbf{F_i}) = f(\mathbf{F_i}) \\ \Leftrightarrow\ &\begin{pmatrix} \mathbf{b^j_1} - \mathbf{b^i_1} \\ \mathbf{b^j_2} - \mathbf{b^i_2} \\ \mathbf{N^j} - \mathbf{N^i} \end{pmatrix} = \begin{pmatrix} \Gamma^{i,1}_{j,1}\mathbf{b^i_1} + \Gamma^{i,2}_{j,1}\mathbf{b^i_2} + A^i_{j,1}\mathbf{N^i} \\ \Gamma^{i,1}_{j,2}\mathbf{b^i_1} + \Gamma^{i,2}_{j,2}\mathbf{b^i_2} + A^i_{j,2}\mathbf{N^i} \\ \Gamma^{i,1}_{j,3}\mathbf{b^i_1} + \Gamma^{i,2}_{j,3}\mathbf{b^i_2} + A^i_{j,3}\mathbf{N^i} \end{pmatrix} \end{aligned} \qquad (12)$$

Also the Equation (12) can be expressed as the relationship between the vertex one of its 1-ring neighbours. The Equation (13) shows the idea:

$$\begin{aligned} &\begin{pmatrix} \mathbf{b^j_1} \\ \mathbf{b^j_2} \\ \mathbf{N^j} \end{pmatrix} = \begin{pmatrix} (1+\Gamma^{i,1}_{j,1}) + \Gamma^{i,2}_{j,1} + A^i_{j,1} \\ (1+\Gamma^{i,1}_{j,2}) + \Gamma^{i,2}_{j,2} + A^i_{j,2} \\ (1+\Gamma^{i,1}_{j,3}) + \Gamma^{i,2}_{j,3} + A^i_{j,3} \end{pmatrix} \begin{pmatrix} \mathbf{b^i_1} \\ \mathbf{b^i_2} \\ \mathbf{N^i} \end{pmatrix} \\ \Leftrightarrow\ &\mathbf{F_j} = \mathbf{T_{ij}}\mathbf{F_i} \end{aligned} \qquad (13)$$

In the equation, $\mathbf{T_{ij}}$ is the transformation matrix between two adjacent frames. Significantly, the transformation matrix is the implicit property between the two discrete frame. Thus, with the discrete frame relationships, when the positional attributes of the geometry mesh change, the transformation relationship between the discrete frame of two neighboring vertices will be changeless. The constrains can be used for preventing the stretching of original deformation methods like Laplacian mesh editing.

4.2 Mesh editing with rotate-invariant frame

With the definition of the discrete frame, now we can construct the pipeline of mesh editing under the rotate-invariant constrains. The mesh reconstruction can be concluded into five steps shown below:

I. Construct the local frame for each vertex.

Traverse all the vertices of the geometry mesh. For each vertex, its local discrete frame is the triplet $(\mathbf{b^i_1}, \mathbf{b^i_2}, \mathbf{N_i})$ shown in the Figure 3.

For the frame construction, we firstly construct a vertex normal. For each vertex V_i, we traverse all of its 1-ring neighboring triangles and do the area weighting sum by considering normals of all the its 1-ring neighboring triangles. For each triangle, its normal comes from the cross product of any its two edges.

Then there are two methods to find the tangent basis on the tangent plane. One way is as [Meyer et al. 2003] described. Get the tangent basis by projecting one neighboring edge to the tangent plane, which can also be seen in 3. The other way is directly use $(y_N, -x_N, 0)$ as the tangent basis, y and x are the coordinates in $N_i\,(x_N, y_N, z_N)$. In this paper, we used the later way which is fast for finding but need to consider the specific case when the vector is paralleling to the Z axis.

Finally, after gotten the normal and one tangent basis of the vertex V_i, the second tangent basis can be calculated by doing cross product of the two known vectors.

II. Change the global Laplacian coordinates of each vertex into local Laplacian coordinates.

Traverse all the vertices of the geometry mesh. For each vertex V_i, calculate its local Laplacian coordinates. Basing the global Laplacian coordinates $\delta^{\mathbf{i}}$ as shown in 3, we can construct the local Laplacian coordinates $\delta^{\mathbf{i}}_{\mathbf{loacal}}$ as shown below:

$$\delta^{\mathbf{i}}_{\mathbf{local}} = \delta^{\mathbf{i}}_{\mathbf{global}} \cdot \mathbf{F_i} = (\delta^{\mathbf{i}}_{\mathbf{global}} \cdot \mathbf{b^i_1},\ \delta^{\mathbf{i}}_{\mathbf{global}} \cdot \mathbf{b^i_2},\ \delta^{\mathbf{i}}_{\mathbf{global}} \cdot \mathbf{N^i}) \qquad (14)$$

III. Built the discrete surface equation system and get the updated local frame.

Traverse all the vertices of the geometry mesh. For each vertex Vi and one of its neighbor V^i_j, we can have the discrete equation system as Equation 13.

Then, we add orientation constrains as the initial conditions. Those positional constrains comes from the anchor points (SCP and HCP) that user defines. Updating the local frame for each vertex by using rotation-invariant coordinates can be conclude as the minimum optimization problem showing below:

$$\begin{cases} E(\mathbf{F}') = \sum\limits_{i,j\in E} \|\mathbf{F_j}' - \mathbf{T_{ij}}\mathbf{F_i}'\|^2 + \sum\limits_{k=1}^{m} \alpha_k \|\mathbf{F_k}' - \mathbf{R_k}\|^2 \\ \mathbf{F}' = \underset{\mathbf{F}'}{arg\,min}(E(\mathbf{F}')) \end{cases} \qquad (15)$$

.

The Equation (15) can be expressed as a matrix format:

$$\begin{pmatrix} \mathbf{G} \\ \mathbf{A_{rank=k}} \end{pmatrix} \mathbf{F}' = \begin{pmatrix} \mathbf{0} \\ \mathbf{R} \end{pmatrix} \tag{16}$$

From Equation (16), $\mathbf{G}$ is the coeffieient matrix referring transformation matrix T_{ij}. $\mathbf{F}'$ is the updated discrete frames of the mesh vertices. $\mathbf{A_{rank=k}}$ is a irregular weighting matrix whose rank is k, which is larger than the number of the lack rank of $\mathbf{G}$. $\mathbf{R}$ is the orientation constrains of frames from anchor points. The detail parameter of the equation can be seen in Equation (17), in which $\mathbf{T_{ij}}$ is the 3×3 transformation matrix related to the local frame triple:

$$\left[\begin{array}{cccc} \sum_{j^0 \in N(0)} \mathbf{T_{0j}} & -1 & \cdots & 0 \\ 0 & \sum_{j^1 \in N(1)} \mathbf{T_{1j}} & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & -1 & \cdots & \sum_{j^n \in N(n)} \mathbf{T_{nj}} \\ \hline 0 & \mathbf{W_2} & \cdots & 0 \\ 0 & \cdots & \mathbf{W_p} & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \mathbf{W_n} \end{array}\right] \left[\begin{array}{c} \mathbf{F_0'} \\ \mathbf{F_1'} \\ \vdots \\ \mathbf{F_n'} \end{array}\right] = \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \hline \mathbf{F_2} \\ \mathbf{F_p} \\ \vdots \\ \mathbf{F_n} \end{array}\right] \tag{17}$$

To get the local frames updating result, the solutions of Equation (16) or Equation (17) are the answer. The matrix equation system can be classified as tremendous linear sparse equation system with over-constrained conditions. Thus the method exist low-speed problem even if we use the least-square method after doing Cholesky decomposition.

IV. Get the new global Laplacian coordinates with the new local frame.

After we got the updated local frame, we can get the new Laplacian coordinates based on the rotate-invariant coordinates. For each vertex V_i the step can be concluded as the formula below:

$$\delta_{\mathbf{global}}^{\mathbf{i}}{}' = \delta_{\mathbf{local}}^{\mathbf{i}} \cdot \mathbf{F_i}' = (\delta_{\mathbf{local}}^{\mathbf{i}} \cdot \mathbf{b_1^i}', \ \delta_{\mathbf{local}}^{\mathbf{i}} \cdot \mathbf{b_2^i}', \ \delta_{\mathbf{local}}^{\mathbf{i}} \cdot \mathbf{N^{i\prime}}) \tag{18}$$

V. Go back to the algorithm of Laplacian mesh editing talked in Section 3.2 and get the deformed mesh.

With the new Laplacian coordinates for each vertices, we can update the positional parameters of the vertices by applying the Equation 6. The updated equation system can be seen below:

$$\begin{pmatrix} \mathbf{L}' \\ \mathbf{I_{rank=k}} \end{pmatrix} \mathbf{V}' = \begin{pmatrix} \delta \\ \mathbf{h} \end{pmatrix} \tag{19}$$

The solution of the Equation 19 can implement the mesh editing without the sensitivity of coordinates rotation. The result can be seen in Section 5.

5 Results

In this section, we provide some basic result of rotate-invariant deformation, which implemented the basic mesh transformation, such as translation, scale and rotation. We also realized some improvement about the GUI for more accurate mesh editing.

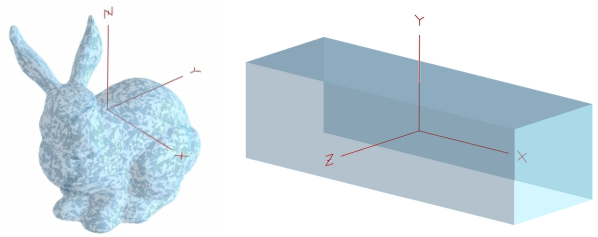


Figure 4: The Cartesian coordinate system for the global space of the Bunny and Cuboid models. Here we set the model with high transparency for better visual explanations.

5.1 Basic Deformations

For the original experiment, we set a pair of anchor points. One is a SCP (marked as the blue point) and the other is a HCP (marked as the red point). For all of those transformations we did not change the positional constrains but the orientation constrains. The Cartesian coordinate system can be seen in Figure 4.

The result of translation can be seen in the Figure 5. The middle column pictures shows the original model in different angle of view in the direction of global axis. The first row shows the stretching in the direction of X axis. The second row shows the stretching in the direction of Z axis. The third row shows the stretching in the direction of Y axis.

The Figure 7 shows the result of rotation and scale. For these two effect, unlike the Laplacian mesh deformation, we need to set orientation constrains with some transformation. The idea can be shown in the Equation 20.

$$\begin{aligned} \mathbf{F_{cons}}' &= \mathbf{T_{frame}} \cdot \mathbf{F_{cons}} \\ &= (\mathbf{T_{frame}} \cdot \mathbf{b_1^{cons}}, \ \mathbf{T_{frame}} \cdot \mathbf{b_2^{cons}}, \ \mathbf{T_{frame}} \cdot \mathbf{N^{cons}}) \end{aligned} \tag{20}$$

For the rotation experiment, we used the Cuboid model. For better visualization, we set different pair of anchor points with different pair of positions between the pictures in the second column in Figure 7. For the scale experiment we used the Bunny model. The scaled SCP point is set on the top of the head of the Bunny model. The result can be seen in the last two columns in Figure 7.

5.2 Improvement for rotate-invariant deformation

We did some improvement of the GUI for a better results of rotate-invariant deformation. There are two major improvements.

The first one is the improvement for anchor points selecting. We increased the accuracy for vertices selecting (which can be seen in Figure 1) and the numbers of selectable SCP and HCP. For the former, we change the selection from a triangle into a vertex. As Figure 6 shows, to get an accurate vertex selection, we change to choose two alternate triangles for the vertex.

The second one is the improvement for the number of selectable anchor points. Considering the calculation speed, our system can set at most 4 HCPs and 4 SCPs. With more numbers of anchor points constrains, our rotation transformation of the rotate-invariant deformation can be more accurate. Taking the Cuboid model as an example, instead of setting only two anchor points, we can set 4 HCPs at the peaks of the X-negative face and 4 SCPs at the peaks of the X-positive face.As Figure 8 shows, the rotation in the direction of Z axis can be seen in the first row and the rotation in the direction of X axis can be seen in the second row. In the experiment result,

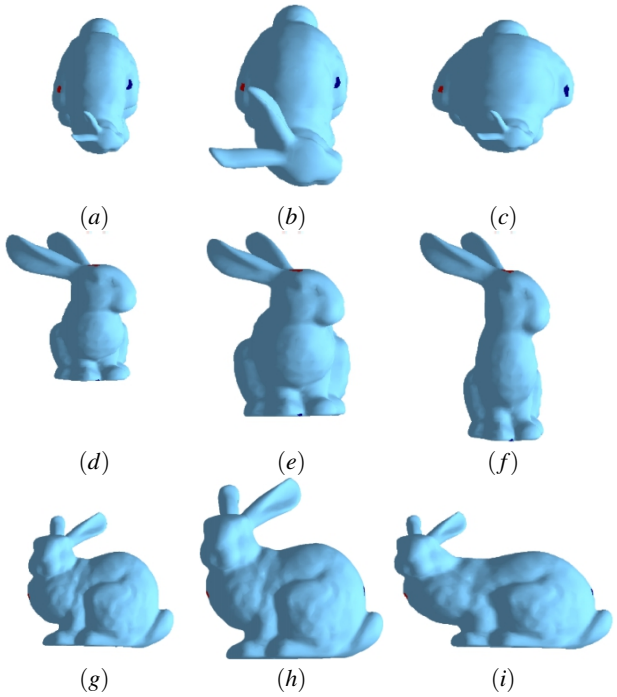


Figure 5: (b), (e) and (h) are the original model in the different angle of view. (a) shows the translate effect when the SCP moves in the negative direction of X axis with 0.3. (c) shows the translate effect when the SCP moves in the positive direction of X axis with 0.3. (d) shows the translate effect when the SCP moves in the positive direction of Z axis with 0.3. (f) shows the translate effect when the SCP moves in the negative direction of Z axis with 0.3. (g) shows the translate effect when the SCP moves in the negative direction of Y axis with 0.3. (i) shows the translate effect when the SCP moves in the positive direction of y axis with 0.3.

.

the red points is marked as the HCPs and the green points is marked as the SCPs.

Additionally, if we combine the improved rotation transformations with the translation and the scale transformations under the law of rotate-invariant mesh editing, we can finally get the result as (a) and (b) showing in Figure 1.

6 Conclusion

6.1 What we did

Our system has realized eligible deformation result under the constrains from rotate-invariant coordinates. The mesh editing tool is able to control the handle points by doing the combinations of translations, rotations and scales. The result nicely protect the details of the geometry model. Compared with the classical Laplasican mesh deformation, our algorithm can update the orientation of Laplacian coordinates besides the positional updating. However, there are also some limitations existing in our system when considering the efficiency of the GUI, the calculation speed of solving the sparse linear equation system, and the incomplete ROI (Region of Interests) selections.

6.2 Limitations for improving directions

The ineffective GUI: Our system can automatically calculate the parameter changes of rotation or translation, which can be seen in (b) in Figure 1, Figure 8, and the first row of Figure 8. However, for

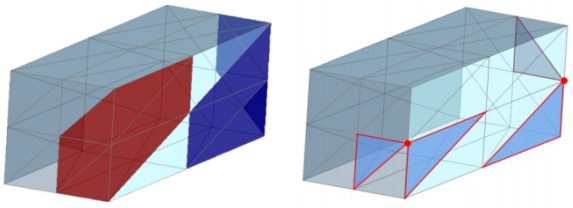


Figure 6: The way to improve the accuracy of vertex selection. The left side method is choosing a mesh triangle then set its vertex with index 0 as the anchor point. The right side one can easily realize vertex selection by choosing two alternate triangles.

the case of screwing or rotating with the non-axis rotation axis, the correspond positional constrain need to be precalculated manually. The inconvenience also means our system do not have a function of handle trace ball, which can calculate the positional displacement of the SCPs corresponding to the local frame rotation.

There is a direction to solve this problem according to the description from [Sorkine et al. 2004]. We can change the error Equation of Laplasican coordinates updating of the last step of rotate-invariant mesh editing into:

$$\begin{cases} E(\mathbf{V}') = \sum_{i=1}^{n} \|\mathbf{T_i}(V_i')\delta_\mathbf{i} - \mathscr{L}(V_i')\|^2 + \sum_{i=m}^{n} \|V_i' - U_i\|^2 \\ \mathbf{V}' = \underset{\mathbf{V}'}{arg\,min}(E(\mathbf{V}')) \end{cases} \tag{21}$$

.

$\mathbf{T_i}$ is the 3×3 linear transformation matrix for the displacement vertex V_i. $\mathbf{T_i}$ can be deviated from the error equation system:

$$\begin{cases} E(\mathbf{T_i}) = \|\mathbf{T_i}V_i - V_i'\|^2 + \sum_{j \in N_1(i)} \|\mathbf{T_i}V_j - V_j'\|^2 \\ \mathbf{T_i}' = \underset{\mathbf{T_i}'}{arg\,min}(E(\mathbf{T_i}')) \end{cases} \tag{22}$$

.

However, by considering the performance of the speed of our system, solving the extra linear systems for handle points with their neighbours will increase the time complexity. Thus this method can be referred once our system can figure out the high time complexity of the solving algorithm of the former two tremendous sparse equation systems.

The high time complexity: Even if consider all the improvement of the GUI, our system still spend too much time for the calculations of complex meshes. For the Bunny model, with 4 HCPs and 2 SCPs as (a) and (b) shown in Figure 1, it costs almost a minutes to get the deformation result. The slow processing speed will directly decrease the user experience especially when they want to see dynamic smooth mesh deformations.

The incomplete ROI strategy: In the stage of now, the deformation result is not perfect if we observe the over-deformed bottom face of the deformed Cuboid model in (d) and (f) in Figure 8. The reason of this results is because we only set several boundary constrain points but not include the points which are inside of the boundary. Thus, instead of setting limited anchor points for the ROI that user defines, we should allow our system to choose discrete faces with some fixed area.

Acknowledgements

This paper is based on the previous paper work named *Linear Rotation-invariant Coordinates for Meshes* [Lipman et al. 2005]. The code got helped from Prof. Eugene Zhang. The total content only used for the learning of the course CS554(Geometry Modeling) in Oregon State University.

References

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. In *ACM SIGGRAPH 2004 Papers*, Association for Computing Machinery, New York, NY, USA, SIGGRAPH '04, 630–634.

KOBBELT, L., CAMPAGNA, S., VORSATZ, J., AND SEIDEL, H.-P. 1998. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, Association for Computing Machinery, New York, NY, USA, SIGGRAPH '98, 105–114.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3 (July), 479–487.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, Springer Berlin Heidelberg, Berlin, Heidelberg, H.-C. Hege and K. Polthier, Eds., 35–57.

SHEFFER, A., AND KRAEVOY, V. 2004. Pyramid coordinates for morphing and deformation. In *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.*, 68–75.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, Association for Computing Machinery, New York, NY, USA, SGP '04, 175–184.

WIKIPEDIA CONTRIBUTORS, 2021. 3d modeling — Wikipedia, the free encyclopedia. [Online; accessed 15-March-2021].

WIKIPEDIA CONTRIBUTORS, 2021. Laplace operator — Wikipedia, the free encyclopedia. [Online; accessed 15-March-2021].

WIKIPEDIA CONTRIBUTORS, 2021. Laplace–beltrami operator — Wikipedia, the free encyclopedia. [Online; accessed 15-March-2021].

WIKIPEDIA CONTRIBUTORS, 2021. Polygon mesh — Wikipedia, the free encyclopedia. [Online; accessed 15-March-2021].

YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2004. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3 (Aug.), 644–651.

ZAYER, R., ROESSL, C., KARNI, Z., AND SEIDEL, H.-P. 2005. Harmonic Guidance for Surface Deformation. *Computer Graphics Forum*.

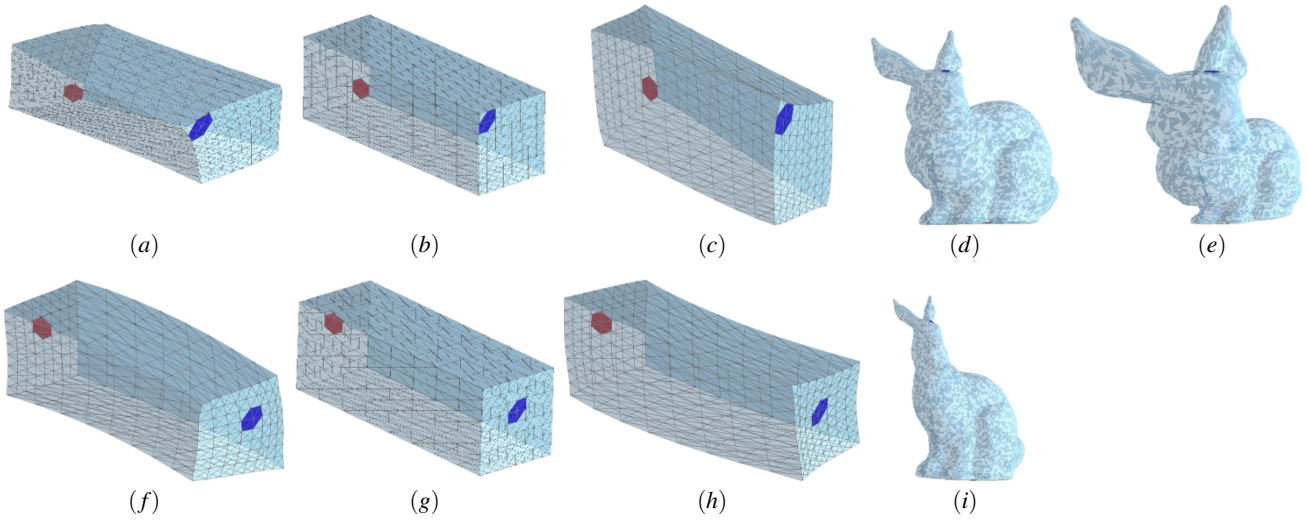


Figure 7: (b), (g) and (d) are the original model in the different angle of view. (a) shows the rotation effect when the SCP moves in the positive direction of X axis with $\frac{\pi}{6}$. (c) shows the rotation effect when the SCP moves in the negative direction of X axis with $\frac{\pi}{6}$. (f) shows the rotation effect when the SCP moves in the negative direction of Y axis with $\frac{\pi}{6}$. (h) shows the rotation effect when the SCP moves in the negative direction of Y axis with $\frac{\pi}{6}$. (e) shows the scale effect when the SCP is enlarged 2 times as the original model. (i) shows the scale effect when the SCP is enlarged 0.5 times as the original model.

.

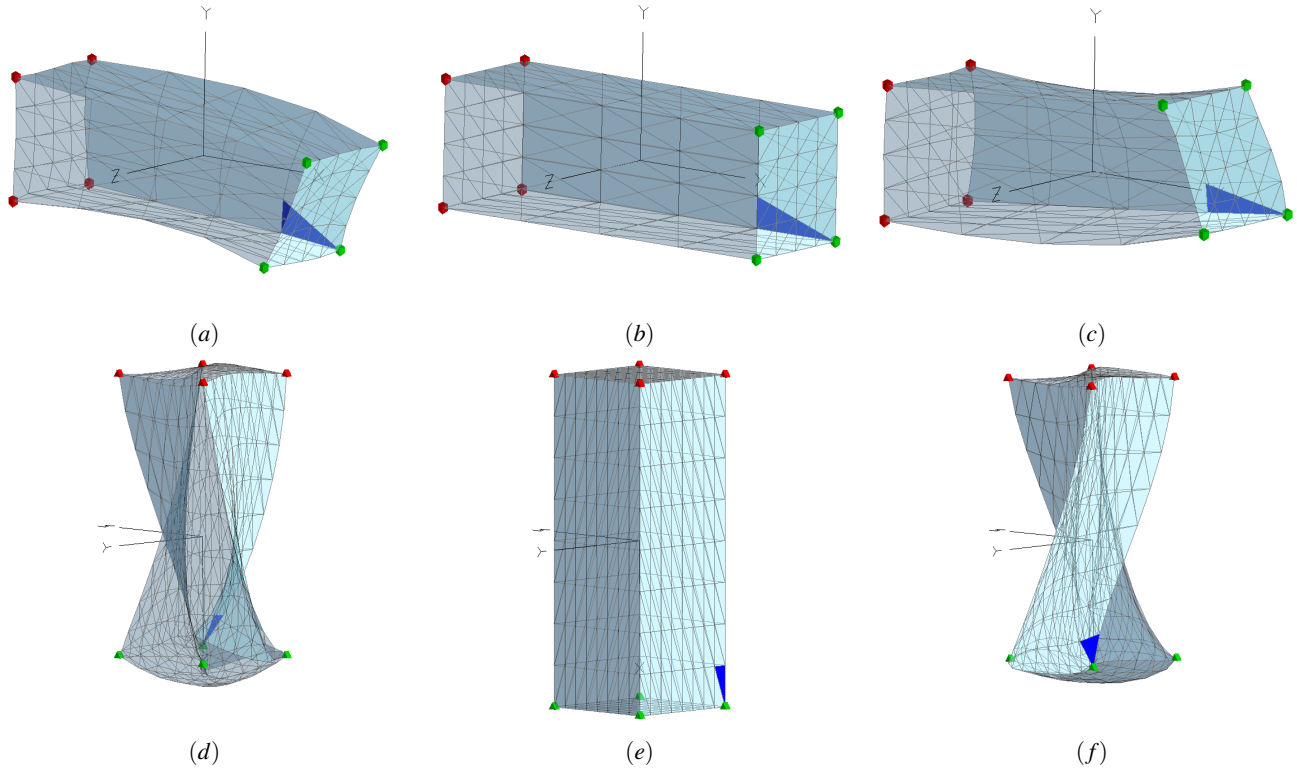


Figure 8: (b) and (e) are the original model in the different angle of view. (a) shows the improved rotation result when the SCPs rotate around the negative direction of Z axis with $\frac{\pi}{6}$. (c) shows the improved rotation result when the SCPs rotate around the positive direction of Z axis with $\frac{\pi}{6}$. (d) shows the improved rotation result when the SCPs rotate around the negative direction of X axis with $\frac{\pi}{2}$. (f) shows the improved rotation result when the SCPs rotate around the positive direction of X axis with $\frac{\pi}{2}$.

.