Acquisition and Processing of 3D Geometry
Project: Volumetric Deformation

Jiacheng Liu and Xinyi Yu
Due date: 28th May. 2018

Introduction

In this project, we attempted to implement cage-based volumetric deformation by using different coordinates, including Mean Value Coordinates and Green Coordinates. To compare the performance of different algorithm, we also programmed simple deformation for the evaluation. The whole project was implemented in C++ programming language by our group. This report will give a description about reference paper and necessary steps included and show related screen shots of results. In the first section, the uses of model and cage in this project will introduced and the overall user-interface of the project will be shown as well. Next, the deformation based on Mean Value Coordinates and Green Coordinates will be described in both paper summary and results evaluation in following sections (algorithms can be found in folder 'Volumetric_Deformation'). Finally, we implemented a simple deformation algorithm as novel section to compare the reformation performance with above two cage-based algorithms (algorithm can be found in folder 'Deformation_without_Normal').

Xinyi and I have clear responsibilities in this project. In the part of cage-based deformation, Xinyi implemented the Green Coordinates deformation and I implemented Mean Value Coordinates deformation, while in the part of simple deformation, Xinyi implemented deformation for tapering and twisting and I programmed deformation for bending operation. Additionally, each of us completed half visualization and half evaluation.

1. Model & Cage

We used two models for testing in this project, which are named 'bunny.off' and 'cow.off'.

For the first model 'bunny.off', we defined a simple cage by ourselves manually for it by using software called Blender and then used Meshlab to convert it to '.off' file. Because the structure for this cage is rather simple, we used it for the single point-control deformation.

For the second model 'cow.off', we found it and its cage on a reference model dataset named TURBOSQUID. Because we cannot use libraries in C++ for mouse handle-drive operation due to computer environment, we modified parts of original cage to define two new cages for target deformation by using Blender for evaluating performance of algorithms for partially deformation. Then all of them were converted into '.off' format for convenient use in later program.

Two models with their cages are shown below.

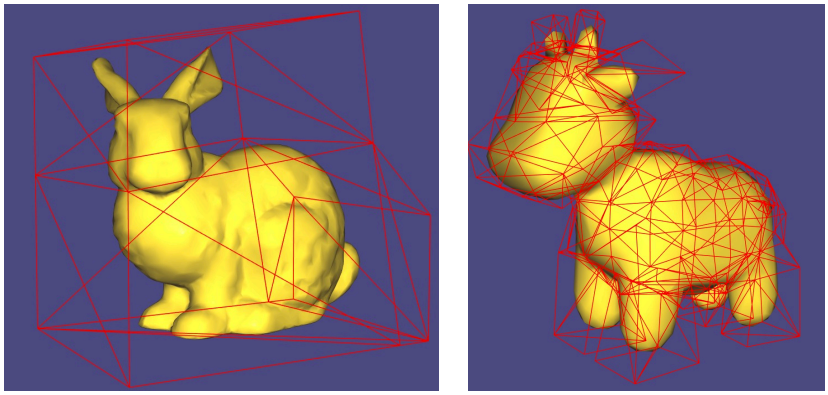


Figure 1: 'bunny.off' (left) and 'cow.off' (right)

In order to evaluate algorithm with convenient operations, we designed below interface together. There are four steps included to achieve the deformation.

- Step 1: Select model
- Step 2: Show original cage
- Step 3: Define target cage. For model 'bunny.off', user can decide move which vertex and corresponded translation matrix. For model 'cow.off', there are two pre-defined deformed cages provided and user can choose one of them to observe the results.
- Step 4: Select deformation mode, including Mean Value Coordinates and Green Coordinates, and observe the deformation results.

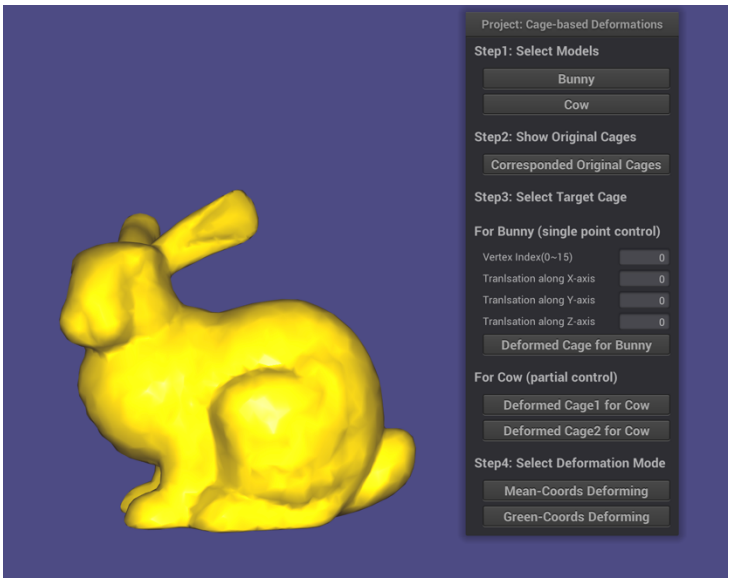


Figure 2: The developed interface

2. Mean Value Coordinates

For the implementation of Mean Value Coordinates, I read *'Mean Value Coordinates for Closed Triangular Meshes'* by Tao. J., Scaott. S. and Joe. W. as the reference and developed the interpolation algorithm based on it.

Paper summary [1]

This paper gave a very detailed description of the Mean Value Coordinates from several aspects, including basic definition and properties, interpolation algorithms for both 2D polygons and 3D meshes and related applications, such as boundary value interpolation, volumetric textures and surface deformation. Finally, it evaluated performance of Mean Value Coordinates in terms of deformed results and computation speed. However, this algorithm does not ensure the shape invariance.

Algorithm description [1]

Given a point v and a closed mesh model, iterate each triangle T in the mesh with compose vertices $\{p_1, p_2, p_3\}$ and their corresponded values $\{f_1, f_2, f_3\}$. Then the stages for mean value interpolation could be summarized as following.

1. Compute the mean vector m

   In this stage, a theorem is used for mean vector m calculation. Given a spherical triangle $\bar{T}$, assume θ_i is the length of the i^{th} edge of the triangle and n_i is the inward normal with unit length to the i^{th} edge. Then the mean vector can be obtained by the integral of the outward normal with unit length over $\bar{T}$,

   $$m = \sum_i \frac{1}{2}\theta_i n_i$$

   Figure 3 gives an illustration for the mean value calculation description above.

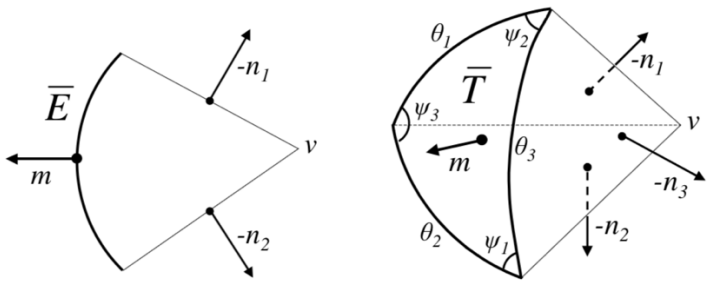


Figure 3: (a) Mean vector m on a circular arc $\bar{E}$ with edge inward unit normal n_i and (b) on a spherical triangle $\bar{T}$ with arc lengths θ_i and face normal n_i

2. Compute the weight w_i

   The relation equation to connect weights w_i with mean vector m can be expressed as below,

   $$\{w_1, w_2, w_3\} = m\{p_1 - v, p_2 - v, p_3 - v\}^{-1}$$

   According to the mean vector m calculated in last stage, now we can compute w_i by applying below equation

$$w_i = \frac{n_i \cdot m}{n_i \cdot (p_i - v)}$$

3. Update the denominator and numerator of interpolant $\hat{f}[v]$ by adding $\sum_i w_i$ and $\sum_i w_i f_i$

   Assume w_i^k and f_i^k are weights and values correspond the edge E_k, then we get

$$\hat{f}[v] = \frac{\sum_k \sum_i w_i^k f_i^k}{\sum_k \sum_i w_i^k}$$

   To obtain correct interpolant $\hat{f}[v]$, we also need to consider two special cases in above procedure. The first one is that the weights w_i computed in second stage may have a zero denominator if the point v lies on the plane that contains the face T. Another case is that triangles with a small projected area, which are dominant type when evaluating mean coordinates on meshes with larger number of triangles. Weights w_i for such triangles are numerically unstable as cancelling of normal n_i that are almost lie on the same plane. We apply different equation for weights calculation,

$$w_i = \frac{\theta_i - \cos[\psi_{i+1}]\,\theta_{i-1} - \cos[\psi_{i-1}]\,\theta_{i+1}}{2\sin[\psi_{i+1}]\sin[\theta_{i-1}]\left|p_i^k - v\right|}$$

   Where ψ_i represents the angle in the spherical triangle $\bar{T}$ and the cosine of it could be obtained as below,

$$cos[\psi_i] = \frac{2\sin[h]\sin[h-\theta_i]}{\sin[\theta_{i+1}]\sin[\theta_{i-1}]} - 1$$

   Where $h = \frac{\theta_1+\theta_2+\theta_3}{2}$.

The pseudo code for the Mean Value Coordinates interpolation could be seen as Figure 4.

```
// Robust evaluation on a triangular mesh
for each vertex p_j with values f_j
    d_j ← ||p_j − x||
    if d_j < ε return f_j
    u_j ← (p_j − x)/d_j
totalF ← 0
totalW ← 0
for each triangle with vertices p_1, p_2, p_3 and values f_1, f_2, f_3
    l_i ← ||u_{i+1} − u_{i−1}||      // for i = 1,2,3
    θ_i ← 2 arcsin[l_i/2]
    h ← (∑ θ_i)/2
    if π − h < ε
        // x lies on t, use 2D barycentric coordinates
        w_i ← sin[θ_i] d_{i−1} d_{i+1}
        return (∑ w_i f_i)/(∑ w_i)
    c_i ← (2 sin[h] sin[h − θ_i])/(sin[θ_{i+1}] sin[θ_{i−1}]) − 1
    s_i ← sign[det[u_1, u_2, u_3]] √(1 − c_i^2)
    if ∃i, |s_i| ≤ ε
        // x lies outside t on the same plane, ignore t
        continue
    w_i ← (θ_i − c_{i+1}θ_{i−1} − c_{i−1}θ_{i+1})/(d_i sin[θ_{i+1}] s_{i−1})
    totalF += ∑ w_i f_i
    totalW += ∑ w_i
f_x ← totalF/totalW
```

Figure 4: Pseudo code for Mean Value Coordinates on 3D triangular mesh [1]

The resulting deformed model satisfies three important properties: interpolation, smoothness and linear precision.

Results and Evaluation

To evaluate the performance of the Mean Value Coordinates, we tested two models with different operations.

For the evaluation in terms of the single-point control for cage, we used 'bunny.off' as the testing model. The coefficients setting for deformed cage as well as deformed model are shown as below.

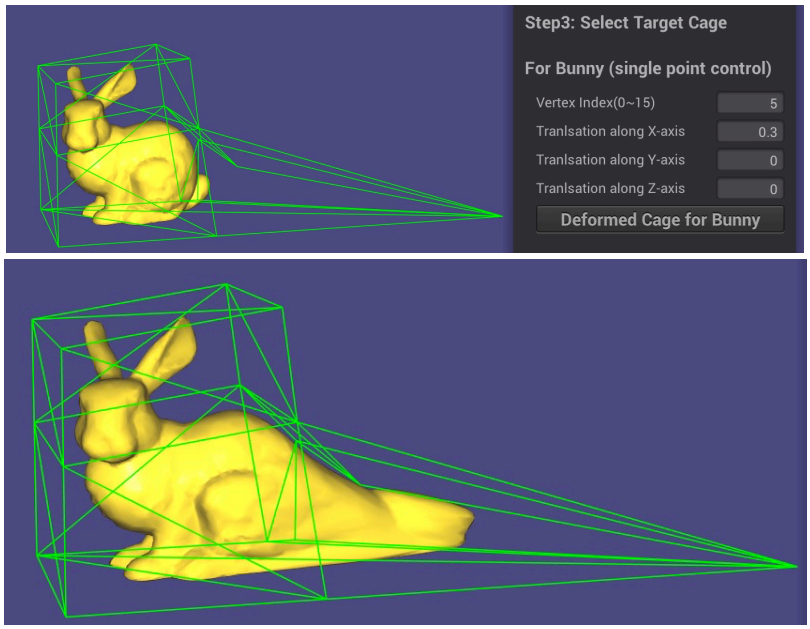
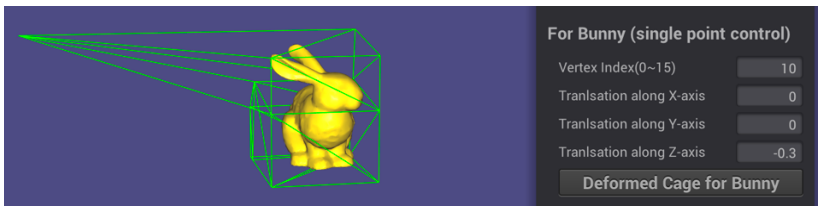


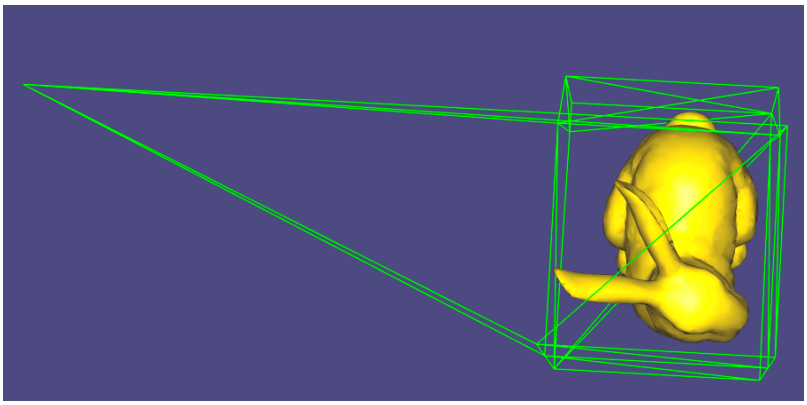
Figure 5: Target cage setting (top) and deformed result (bottom)

From this case, we can find that the tail and backside of bunny model was stretched a lot in the direction that we stretched the cage, which corresponds to the part of target cage that close to tail was translated in x-axis direction. It could be observed that the shape other part of bunny almost does not change and remain the original structure.

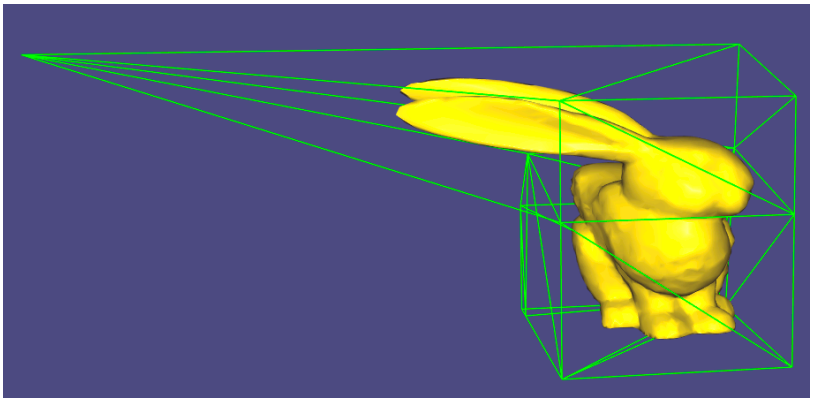
To evaluate the performance with details, we used another case to translate another vertex of cage and the comparison between original and deformed model from different views are shown as Figure 6.



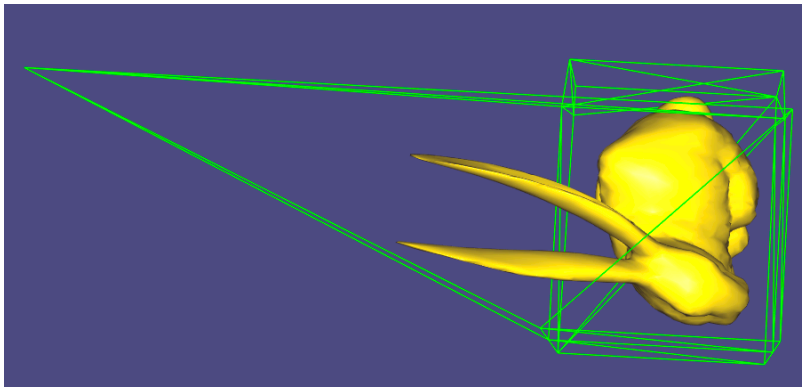
(a) Target cage setting and original model from left view



(b) Original model from top view



(c) Deformed model from left view

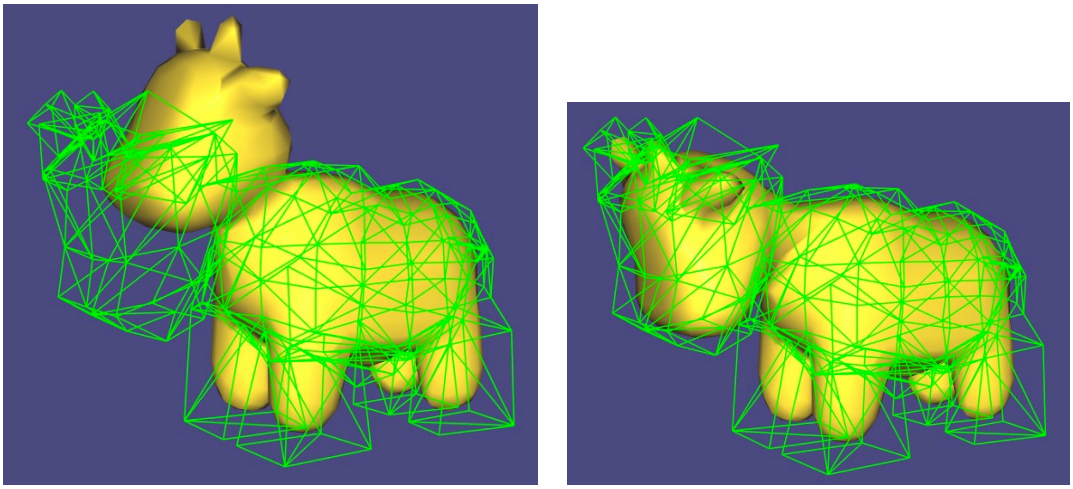


(d) Deformed model from top view

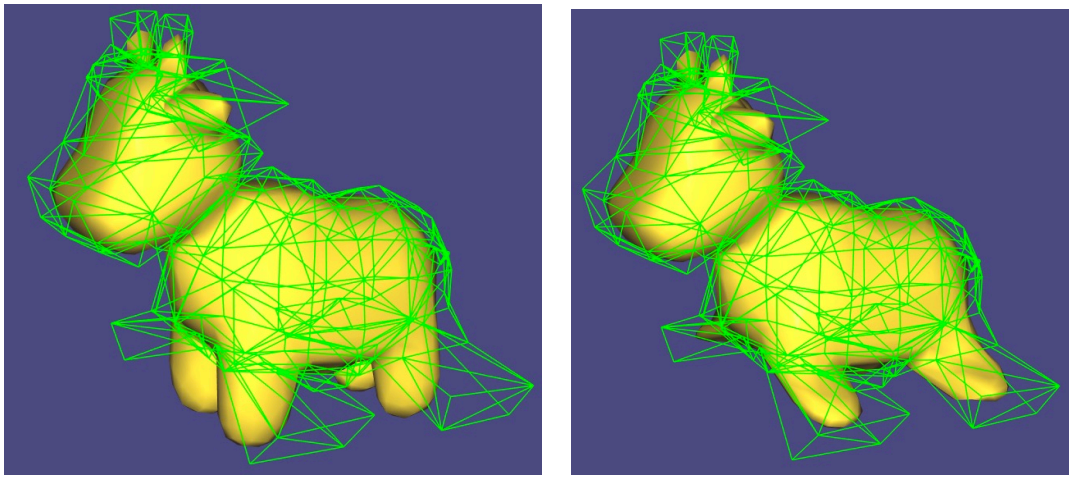
Figure 6: Comparison for cage single-point control

From second case, we can observe that by stretching outward the cage close to the ear part, the ear of the bunny model was stretched in the same direction during the deformation. Meanwhile, the other parts of model were stretched as well in the defined direction with different slight degrees. The closer to the translated vertex, the higher degree of stretches occurs. This also shows that the algorithm of Mean Value Interpolation does not remain shape invariance.

For the evaluation in terms of cage partial control, we used 'cow.off' as the testing model and pre-defined two target cages by using Blender. The comparison between original and deformed models for two cases can be found as Figure 7.



(a) Original (left) and deformed model (right)



(b) Original (left) and deformed model (right)

Figure 7: Comparison for cage partial control

In the first case, the deformed the cow model became look down with the modified cage and all other parts keep the original structure. In the second case, the four legs of cow model moved upward with the transformed cages and they are become flatter compared to original cylinder shape due to the cuts area of modified cage became smaller during the editing. As the first case, almost other parts of the deformed model do not change.

3. Green Coordinates

For the development of Green Coordinates, the reference paper is *'Green Coordinates'* by Yaron. L., David. L. and Danial. C. and Xinyi implemented the interpolation algorithm based on it.

Paper summary [2]

This paper introduced the Green Coordinates or closed polyhedral cages, which are motivated by Green's third integral identity and respect both the vertices and faces orientation of the cage. The closed-form expressions for coordinates were developed for both 2D conformal mappings and 3D quasi-conformal mappings. Finally, this paper compared the deformation preformation of Green Coordinates with Mean Value Coordinates and Harmonic Coordinates to indicate that the achievement of shape-preserving could be kept with both computational speed and simplicity.

Algorithm description [2]

Assume the cage is an oriented simplicial surface (i.e., 2D polygon, 3D triangular mesh) $P = (\mathbb{V}, \mathbb{T})$, where $\mathbb{V} = \{v_i\}_{i \in I_{\mathbb{V}}} \subset \mathbb{R}^d$ are the vertices and $\mathbb{T} = \{t_j\}_{j \in I_{\mathbb{T}}}$ are simplicial face elements. The general framework for defining the coordinates of each interior point η can be expressed as a linear combination,

$$\eta = F(\eta; P) = \sum_{i \in I_{\mathbb{V}}} \phi_i(\eta) v_i + \sum_{j \in I_{\mathbb{T}}} \psi_j(\eta) n(t_j)$$

Where $n(t_j)$ represents the outward unit normal to the oriented simplicial face t_j. Thus, the deformation could be expressed by a deformed cage P' as below,

$$\eta \longmapsto F(\eta; P') = \sum_{i \in I_{\mathbb{V}}} \phi_i(\eta) v_i' + \sum_{j \in I_{\mathbb{T}}} \psi_j(\eta) s_j n(t_j')$$

Where v_i' and t_j' are vertices and faces of P' respectively and scaling factors $\{s_j\}_{j \in I_{\mathbb{T}}}$ is rather important to achieving shape-preserving. The coordinate function ϕ and ψ can be expressed with following functions,

$$\phi_i(\eta) = \int_{\xi \in N\{v_i\}} \Gamma_i(\xi) \frac{\partial G(\xi, \eta)}{\partial n(\xi)} d\sigma_\xi \quad i \in I_{\mathbb{V}}$$

$$\psi_j(\eta) = -\int_{\xi \in t_j} G(\xi, \eta) d\sigma_\xi \qquad j \in I_{\mathbb{T}}$$

Where Γ_i is the piecewise-linear hat function defined on $N\{v_i\}$, which is one at v_i and zero at all other vertices in the 1-ring and linear on each face and $N\{v_i\}$ is the union of all faces in the 1-ring neighborhood of vertex v_i. $G(\cdot,\cdot)$ denotes the fundamental solution of the Laplacian equation in $\mathbb{R}^d$.

The scaler factors in 3D triangular meshes could be expressed by vectors u, v for original face t_j and corresponding vectors u', v' for transformed face t_j',

$$s_j = \frac{\sqrt{\|u'\|^2\|v\|^2 - 2(u' \cdot v')(u \cdot v) + \|u'\|^2\|v\|^2}}{\sqrt{8}area(t_j)}$$

The pseudo code for 3D Green Coordinates interpolation could be seen as Figure 8.

Input: cage $P = (\mathbb{V}, \mathbb{T})$, set of points $\Lambda = \{\boldsymbol{\eta}\}$
Output: $3D$ GC $\phi_i(\boldsymbol{\eta}), \psi_j(\boldsymbol{\eta}), i \in I_{\mathbb{V}}, j \in I_{\mathbb{T}}, \boldsymbol{\eta} \in \Lambda$
/* Initialization */
set all $\phi_i = 0$ and $\psi_j = 0$
/* Coordinate computation */
foreach *point* $\boldsymbol{\eta} \in \Lambda$ **do**
  foreach *face* $j \in I_{\mathbb{T}}$ *with vertices* $\boldsymbol{v}_{j_1}, \boldsymbol{v}_{j_2}, \boldsymbol{v}_{j_3}$ **do**
    foreach $\ell = 1, 2, 3$ **do**
      $\boldsymbol{v}_{j_\ell} := \boldsymbol{v}_{j_\ell} - \boldsymbol{\eta}$
    $\boldsymbol{p} := (\boldsymbol{v}_{j_1} \cdot \boldsymbol{n}(t_j))\boldsymbol{n}(t_j)$
    foreach $\ell = 1, 2, 3$ **do**
      $s_\ell :=$
      $sign\left(\left((\boldsymbol{v}_{j_\ell} - \boldsymbol{p}) \times (\boldsymbol{v}_{j_{\ell+1}} - \boldsymbol{p})\right) \cdot \boldsymbol{n}(t_j)\right)$
      $I_\ell := \mathbf{GCTriInt}(\boldsymbol{p}, \boldsymbol{v}_{j_\ell}, \boldsymbol{v}_{j_{\ell+1}}, 0)$
      $II_\ell := \mathbf{GCTriInt}(0, \boldsymbol{v}_{j_{\ell+1}}, \boldsymbol{v}_{j_\ell}, 0)$
      $\boldsymbol{q}_\ell := \boldsymbol{v}_{j_{\ell+1}} \times \boldsymbol{v}_{j_\ell}$
      $\boldsymbol{N}_\ell := \boldsymbol{q}_\ell / \|\boldsymbol{q}_\ell\|$
    $I := -\left|\sum_{k=1}^{3} s_k I_k\right|$
    $\psi_j(\boldsymbol{\eta}) := -I$
    $\boldsymbol{w} := \boldsymbol{n}(t_j) I + \sum_{k=1}^{3} \boldsymbol{N}_k II_k$
    if $\|\boldsymbol{w}\| > \epsilon$ **then**
      foreach $\ell = 1, 2, 3$ **do**
        $\phi_{j_\ell}(\boldsymbol{\eta}) := \phi_{j_\ell}(\boldsymbol{\eta}) + \frac{\boldsymbol{N}_{\ell+1} \cdot \boldsymbol{w}}{\boldsymbol{N}_{\ell+1} \cdot \boldsymbol{v}_{j_\ell}}$
  end
end

Procedure **GCTriInt**($\boldsymbol{p}$,$\boldsymbol{v}_1$,$\boldsymbol{v}_2$,$\boldsymbol{\eta}$)
$\alpha := \cos^{-1}\left(\frac{(\boldsymbol{v}_2 - \boldsymbol{v}_1) \cdot (\boldsymbol{p} - \boldsymbol{v}_1)}{\|\boldsymbol{v}_2 - \boldsymbol{v}_1\| \|\boldsymbol{p} - \boldsymbol{v}_1\|}\right)$
$\beta := \cos^{-1}\left(\frac{(\boldsymbol{v}_1 - \boldsymbol{p}) \cdot (\boldsymbol{v}_2 - \boldsymbol{p})}{\|\boldsymbol{v}_1 - \boldsymbol{p}\| \|\boldsymbol{v}_2 - \boldsymbol{p}\|}\right)$
$\lambda := \|\boldsymbol{p} - \boldsymbol{v}_1\|^2 \sin(\alpha)^2$
$c := \|\boldsymbol{p} - \boldsymbol{\eta}\|^2$
foreach $\theta = \pi - \alpha, \pi - \alpha - \beta$ **do**
  $S := \sin(\theta) \quad ; \quad C := \cos(\theta)$
  $I_\theta := \frac{-sign(S)}{2}\left[2\sqrt{c}\tan^{-1}\left(\frac{\sqrt{c}C}{\sqrt{\lambda + S^2 c}}\right) + \sqrt{\lambda}\log\left(\frac{2\sqrt{\lambda}S^2}{(1-C)^2}\left(1 - \frac{2cC}{c(1+C) + \lambda + \sqrt{\lambda^2 + \lambda c S^2}}\right)\right)\right]$
return $\frac{-1}{4\pi}\left|I_{\pi-\alpha} - I_{\pi-\alpha-\beta} - \sqrt{c}\beta\right|$

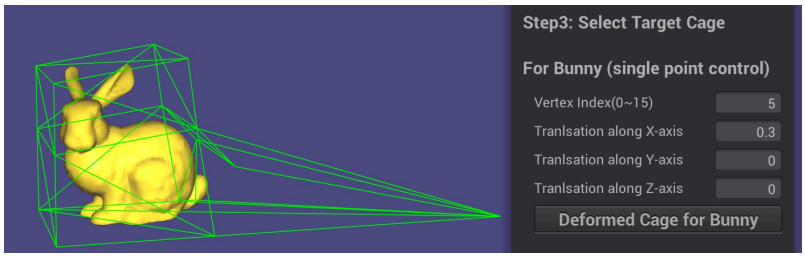
Figure 8: Pseudo code for 3D Green Coordinates

The deformed results should be shape-preserving, including linear reproduction, translation invariance, rotation and scale invariance, shape preservation and smoothness.

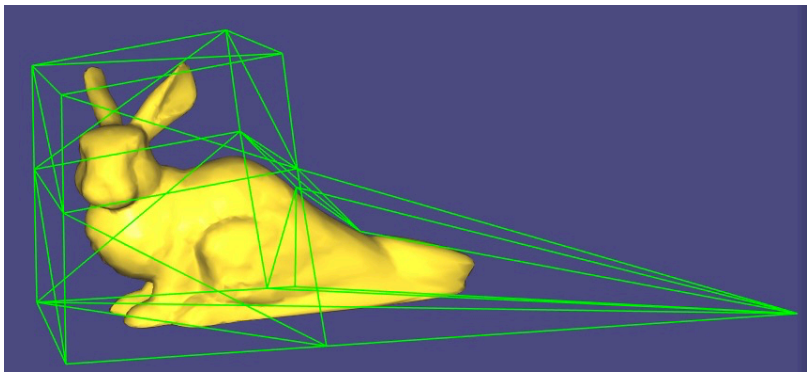
Results and evaluation

Similar as last section, we used two models, 'bunny.off' and 'cow.off' to evaluate the deformation performance of Green Coordinates.

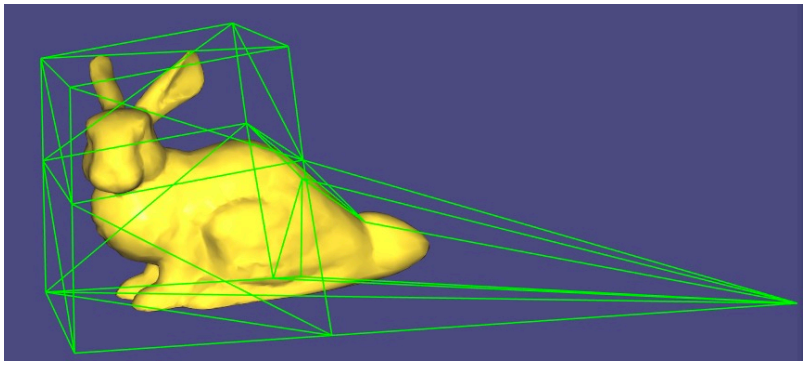
For the evaluation in terms of the single-point control for cage, we used ‘bunny.off’ as the testing model. To clearer observe the shaper preserving property of Green Coordinates, I will analysis results based on the comparison with Mean Value Coordinates. The coefficients setting for deformed cage as well as the comparison of resulting models deformed by Mean Value Coordinates and Green Coordinates are shown as below.



(a) Original model and deformation setting



(b) Mean Value Coordinates

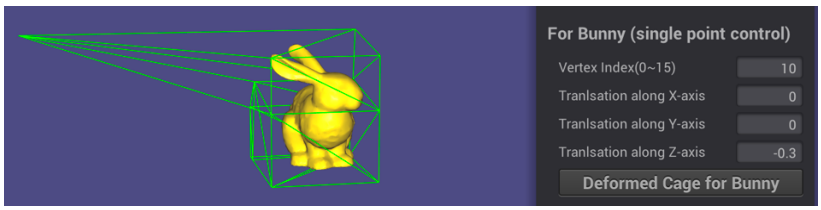


(c) Green Coordinates

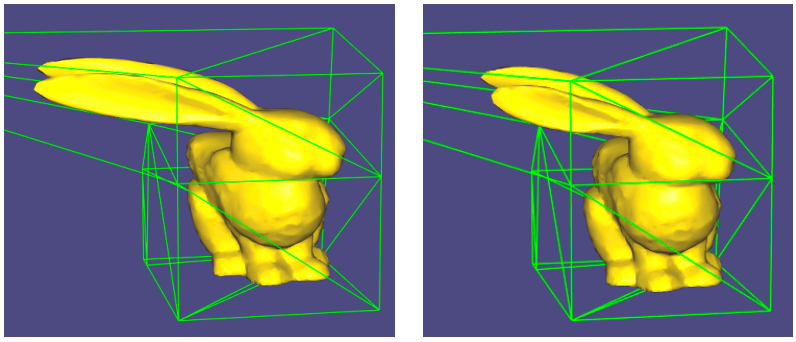
Figure 9: Comparison between Mean Value Coordinates and Green Coordinates

In the first case, we can find that the deformed tail of bunny model based on Green Coordinates in Figure 9(c) keeps its original spherical shape though it is stretched in the direction that cage vertex translates. As the comparison, the tail of deformed model based on Mean Value Coordinates in Figure 9(b) shows some distortion during the stretch and we can observe that some serration at the top of the tail.

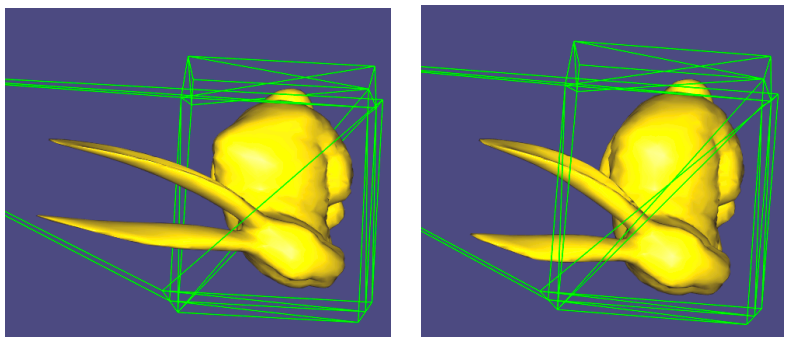
To evaluate the performance with more details, we used another case to translate another vertex of cage and the comparison of deformed model between Mean Value Coordinates and Green Coordinates from different views are shown as Figure 10.



(a) Original model and deformation setting



(b) Left view

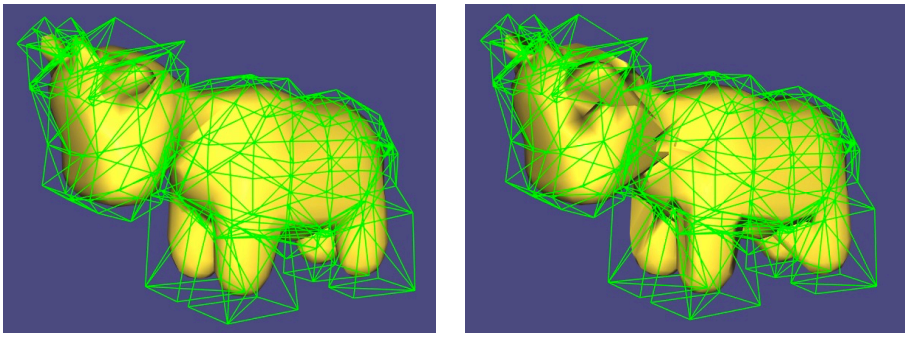


(c) Top view

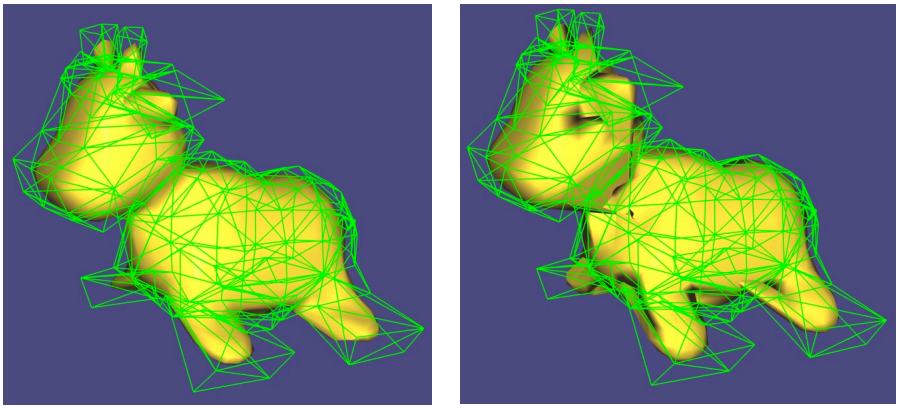
Figure 10: Comparison between Mean Value Coordinates (left) and Green Coordinates (right) for cage single-point control

From Figure 10, we can observe that the deformation based on Green Coordinates is smoother than that based on Mean Value Coordinates. As we mentioned in last section, almost all parts of bunny model are slightly stretched with different degrees in addition to ear in the deformation based on Mean Value Coordinates, while this situation is improved much in the deformation based on Green Coordinates. It could be observed that there is less shape distortion occurs during the deformation based on Green Coordinates. Results in this case provided support that the shape-preserving property of Green Coordinates limits the shape distortion and keeps the model smoothness at the same time. This is because Green Coordinates use both vertices position and face orientation (normal) to define new coordinates of deformed model, while Mean Value Coordinates only consider to get new coordinates by using linear translation of vertices set with weights.

For the evaluation in terms of cage partial control, we used 'cow.off' as the testing model and pre-defined two target cages by using Blender. The comparison of deformation between Mean Value Coordinates and Green Coordinates could be seen in Figure 11.



(a) Case 1



(b) Case 2

Figure 11: Comparison between Mean Value Coordinates (left) and Green Coordinates (right) for cage partial control

From Figure 11(b), we also can find that legs of deformed cow model based on Green coordinates remains smooth cylinder shape, while that based on Mean Value Coordinates become flat much. However, some artifacts occur in the resulting model deformed based on Green Coordinates. In addition to implementation problem (unfortunately we did not find the problem in coding), another possible reason is that the cage mesh used for cow model is too complex, which includes some edges cross the model to connect vertices which are not neighbors with each other. In the further improvement, we can try to remove such edges and only keep edge structure that composed by neighbored vertices.

In addition, to compare the computational time between two algorithms, we record the average time cost of two algorithms for different models. The recorded time cost could be seen as following table.

Table 1: Time cost comparison

Model	**Mean Value Coordinates**	**Green Coordinates**
Bunny	0.086973s	0.126717s
Cow1	0.163278s	0.262639s
Cow2	0.16069s	0.261013s

From the above table, we can find that Mean Value Coordinates have faster speed than Green Coordinates, though both of them work with high efficiency. For model with higher complexity, deformation costs more time.

4. Voxel part: Simple Deformation

As the voxel part of this project, our group implemented pointwise simple deformation. There are three transform operations provided, including tapering, twisting and bending. The designed interface could be seen as below.

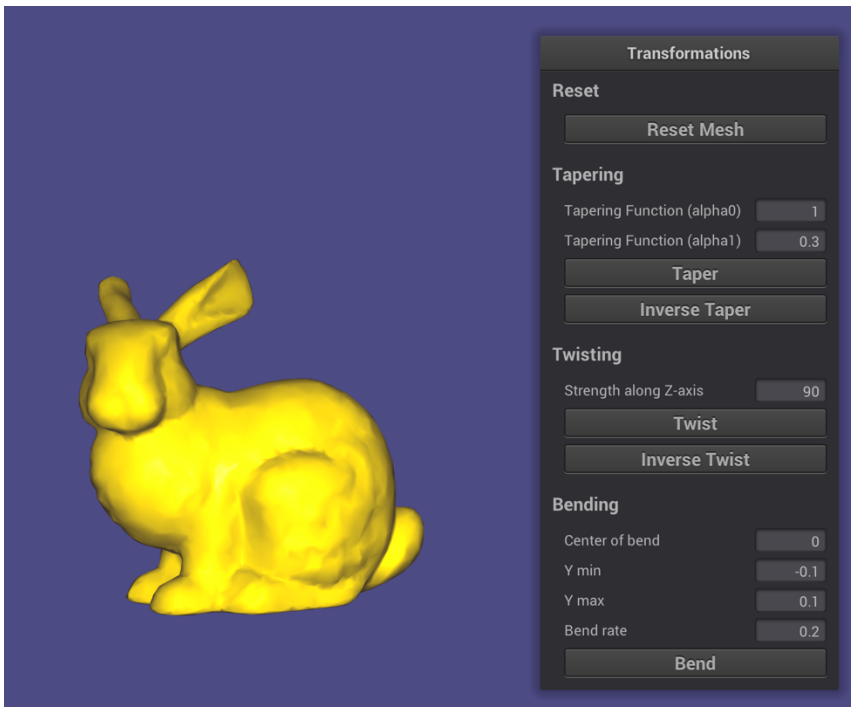


Figure 12: Interface for pointwise deformation

Tapering

The tapering in this project was defined in xy plane. Assume the original point *P(x, y, z)* is deformed as *P'(x',y',z')*. The related computation for forward tapering could be expressed as below,

$$r = \alpha_0 + z\alpha_1$$
$$x' = rx. \quad y' = ry$$

Where α_0 and α_1 are tapering factors.

The expression for inverse tapering are

$$x' = \frac{x}{r}. \quad y' = \frac{y}{r}$$

Twisting

The twisting operation in this project was defined in xy plane as well. The forward twisting calculation can be expressed as below,

$$\theta = scaler \times z$$
$$x' = xcos\theta - ysin\theta$$

$$y' = xsin\theta + ycos\theta$$

The inverse twisting could be expressed as below,

$$x' = xcos\theta + ysin\theta$$
$$y' = -xsin\theta + ycos\theta$$

In this part, user is allowed to define scaler (the twisting strength) in z-axis.

Bending

The bending operation is defined in yz plane in this project. For a defined range y_{min} and y_{max}, intermediate coefficients $\hat{y}$ and θ can be expressed as

$$\hat{y} = \begin{cases} y_{min} & y < y_{min} \\ y & y_{min} \le y \le y_{max} \\ y_{max} & y > y_{max} \end{cases}$$

$$\theta = k \times (\hat{y} - y)$$

Then we can get the deformed coordinates y' and z',

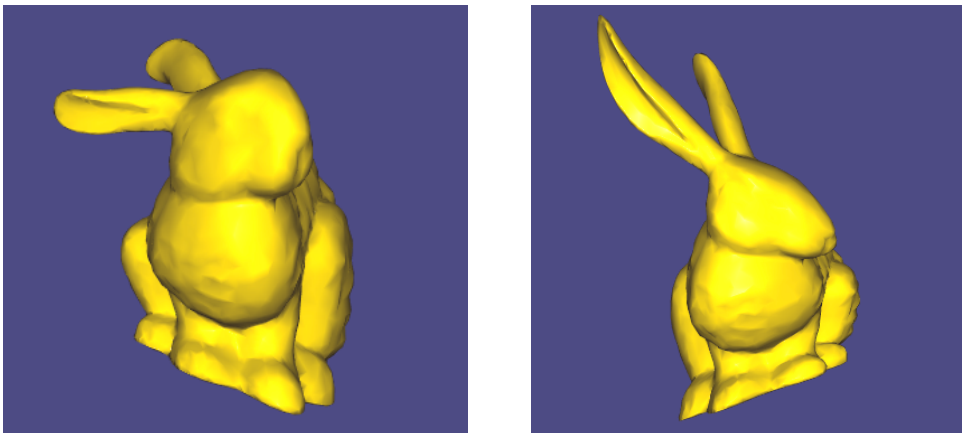
$$y' = \begin{cases} -\sin\theta\left(z - \frac{1}{k}\right) + y + \cos\theta\,(y - y_{min}) & y < y_{min} \\ -\sin\theta\left(z - \frac{1}{k}\right) + y & y_{min} \le y \le y_{max} \\ -\sin\theta\left(z - \frac{1}{k}\right) + y + \cos\theta\,(y_{max} - y) & y > y_{max} \end{cases}$$

$$z' = \begin{cases} \cos\theta\left(z - \frac{1}{k}\right) + \frac{1}{k} + \sin\theta(y - y_{min}) & y < y_{min} \\ \cos\theta\left(z - \frac{1}{k}\right) + \frac{1}{k} & y_{min} \le y \le y_{max} \\ \cos\theta\left(z - \frac{1}{k}\right) + \frac{1}{k} + \sin\theta\,(y_{max} - y) & y > y_{max} \end{cases}$$

Results and evaluation

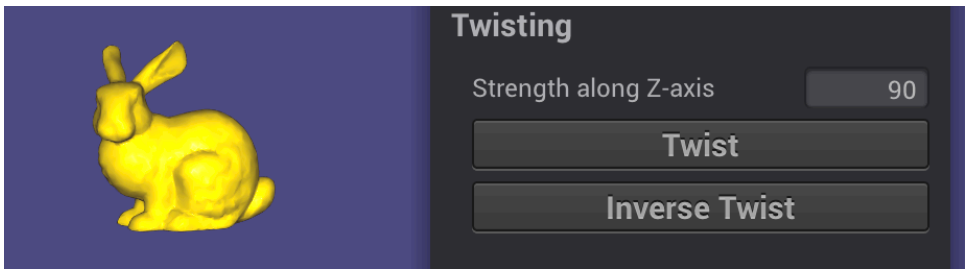
The corresponded deformed results for different operations could be seen as follows.

(a) Original model and tapering setting

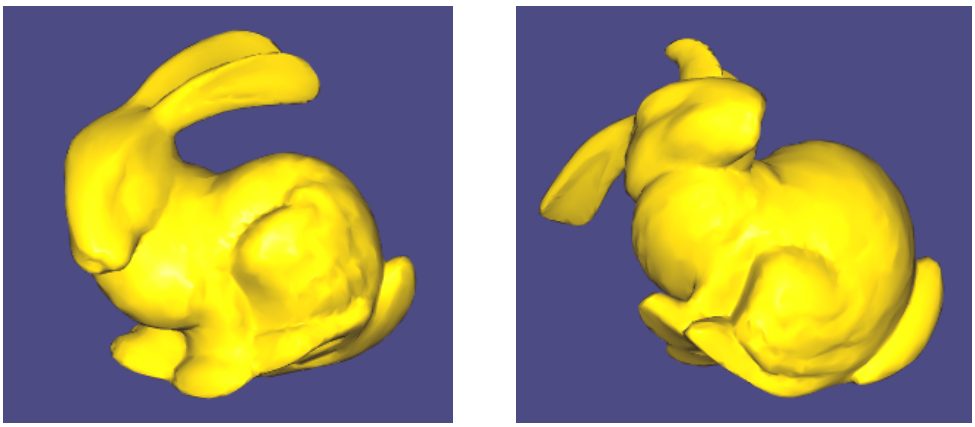


(b) Tapering (left) and inverse tapering (right)

Figure 13: Tapering

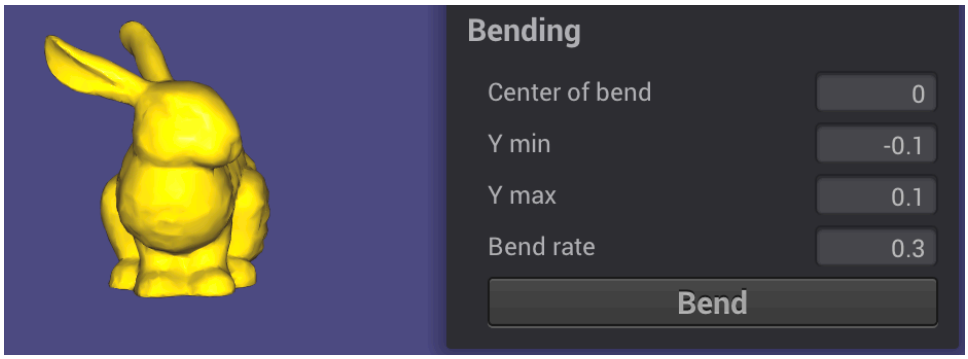


(a) Original model and Twisting setting

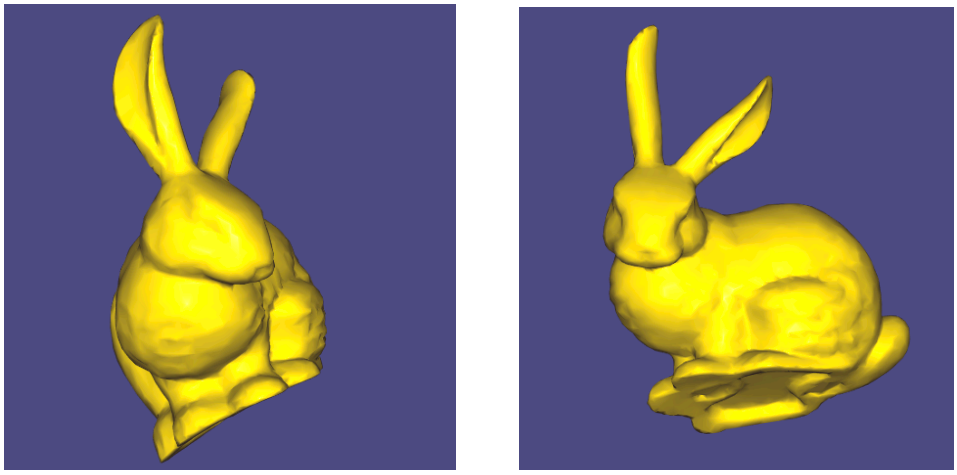


(b) Twisting (left) and inverse twisting (right)

Figure 14: Twisting



(a) Original model and bending setting



(b) Bending results from left view (left) and forward view (right)

Figure 15: Bending

From above results, we can find that the pointwise deformation works well and the bunny was deformed as expected. This algorithm is rather simple to understand and implement. However, compared with cage-based deformation, this algorithm only can achieve global deformation. Local deformation is hard to realize for current developed one as too much coefficients are required to be defined in details. Therefore, this pointwise deformation algorithm lacks flexibility.

Conclusion

In this project, our group implemented cage-based volumetric deformation algorithms based on Mean Value Coordinates and Green Coordinates with clear split of responsibilities. The algorithm works well and most of results are output as expected though some artifacts occur in one of model deformation in Green Coordinates and we will continue to try to use possible solution to overtake it. The comparison between two algorithms were given as well in this report. Finally, we developed a pointwise deformation as the voxel part.

References

[1] Tao. J., Scaott. S. and Joe. W. '*Mean Value Coordinates for Closed Triangular Meshes*'.

[2] Yaron. L., David. L. and Danial. C. *'Green Coordinates'*.