COMPGV18 Project 1

Student Name: Yifei Gao Student ID: 13028702

Paper Summary:

Barr's [1] paper in 1984 suggest methods of local and global deformation of solid primitives, where a global deformation is a mathematical function that explicitly modifies the global coordinates of the vectrices in a mesh, and local deformation only modifies the tangent space of the solid. The manipulations of the tangent vectors is calculated by multiplication of the Jacobian matrix of the transformation function. The Jacobian matrix J is calculated by taking partial derivative of the transformation function $X' = F(X)$, which is calculated by the follow:

$$J(X) = [\, J_1(X), J_2(X), J_3(X) \,] = [\, \frac{\delta F(X)}{\delta x_1}, \frac{\delta F(X)}{\delta x_1}, \frac{\delta F(X)}{\delta x_1}]$$

He also suggested that the tangent vector after transformation is the same as the product of the old tange vector with the Jacobian matrix, given a surface $X = X(u, v)$ then:

$$\frac{\delta X'}{\delta u} = J(X) \frac{\delta X}{\delta u}$$

Another important geometry feature of the surface, the normal vector N could be calculated by

$$N' = det(J) J^{-1T} N$$

Since we usually only care about the direction of the normal, the determinant of the Jacobian matrix is not usually calculated in practical, but based on calculus we know its determinant carries information about the local volume ratio at each point.

They have listed 4 examples of deformation and their corresponding Jacobian Matrix of the deformation, scaling, tapering, twisting and bending. The figure below shows example of each type of deformation:

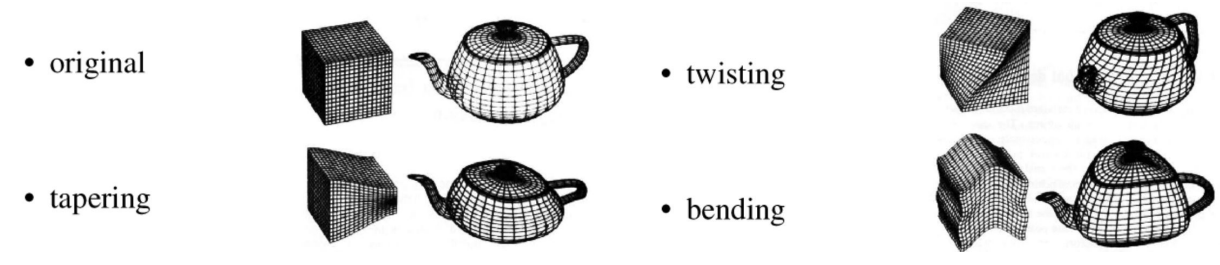

Figure 1: Example of deformation using Jacobian Matrix

The pros of this method are:

1. It provides more feasible tools for generating complicated shapes which is hard to achieve using simple operations like rotation, transform, cut back, boolean, etc.
2. The normal vector of the object after transformation can be calculated using its original mesh and the Jacobian matrix.

Cons of this method:

1. There are only limited amount of specified deformation we could apply to the object, more general deformation have to be done by combination of them.

The limitation of this method lead to development of more generalized deformation method such as FFD (Free-form Deformation) and variation of it, such as EFFD and lattice based FFD. The main idea of such method is to avoid directly manipulated the complicated mesh itself, but embed it into a free form space (i.e A bounding box), where we can manipulates the space instead and the mesh change along with the space it's in. The idea of FFD is first suggested by Sederberg's paper in 1986[2].

Volumetric Deformation follows the same idea as FFD, where the space we embed the mesh into is a low resolution control mesh. The general process of a cage-based deformation is:

1. Construct the cage (crude version of the input model) of the model.(given control mesh P , vertices count $|P| = N$, Model mesh X ,vertices count $|X| = M$, then M>>N)
2. Computes the coordinates of each vertices in the model w.r.t the cage element, where:
$$x = \sum_{i=0}^{n} w_i(x)\, p_i$$
3. Edit the cage element to deform a new cage P'.
4. Smoothly interpolates the model's new vertices position based on P', where
$$x' = \sum_{i=0}^{n} w_i(x)\, p_i'$$

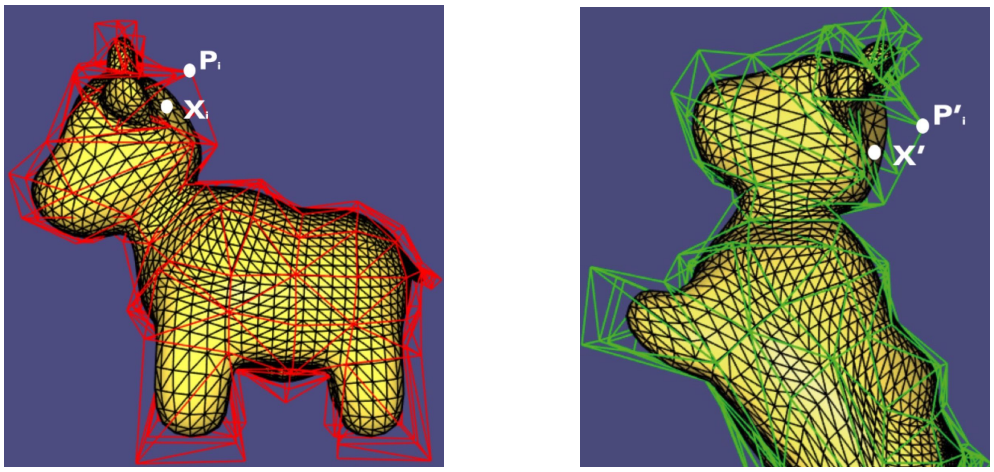

Figure 2: Left: Raw cage with mesh point, RIght: Deformed cage with mesh point

There lots of schemes existing for constructing the coordinate function $w_i(x)$, the commonly used ones are: Mean value coordinates (MVC) [3], Harmonic coordinates and green coordinates.

MVC can be treated as a more generalized version of barycentric coordinates by projecting the triangle on to a unit sphere. In a 2D case we can show it as follow:
Given a point x, and a edge $T = \{v_1, v_2\}$. Placing a unit circle at x then the edge T can projects to an arc $\overline{T}$.

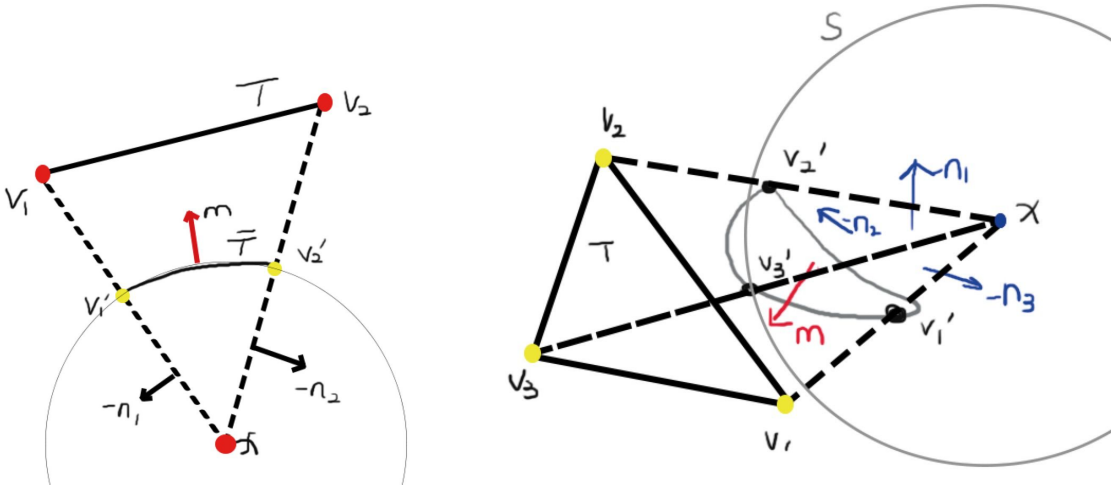

Figure 3: Mean vector on a arc. Right: projection of the triangle on unit sphere

The mean vector $m = u_1^T(v_1 - x) + u_2^T(v_2 - x) = n_1 + n2$, is the integral of the outward unit normal, but as we know the integral of outward unit normal over the whole unit circle is zeros. Hence we know the weight $w_i = \sum u_i^T \; s.t \; T : v_i \in T$ is homogenous. Extending this idea to 3D case, we project the Triangle $T = \{v_1, v_2, v_3\}$ onto a unit sphere and find out the weight using the same approach but with the unit face normals.

The problem with MVC is that it's not invariant to affine transformation, as the coordinate function gives vertex-based basis and it can also gives negative weight for non-convex input model. But as it's simple to implement and can smoothly interpolate non-convex coordinates, it's still commonly used coordinate function in cage-based deformations softwares.

Green's coordinates on the other hand solve this problem by taking into account of the contribution of the cage's face, the formation of the new vertex position w.r.t the deformed cage elements, where

$$x' = \sum_{i=0}^{n} w_i(x)\, p_i' + \sum_{j=0}^{k} h_j(x)\, n_j'$$

Such that $h(x)$ is the weighting function of the cage face normal and n_j is the cage face normal.

Paper implementation and Evaluation:

I have implemented the MVC algorithm suggested by the Sederberg's paper[2], with given model mesh and the cage mesh. I have manipulates the cage mesh using mesh editing software to observer the performance of the algorithm.

Here is a few test result of the deformation with different cages:

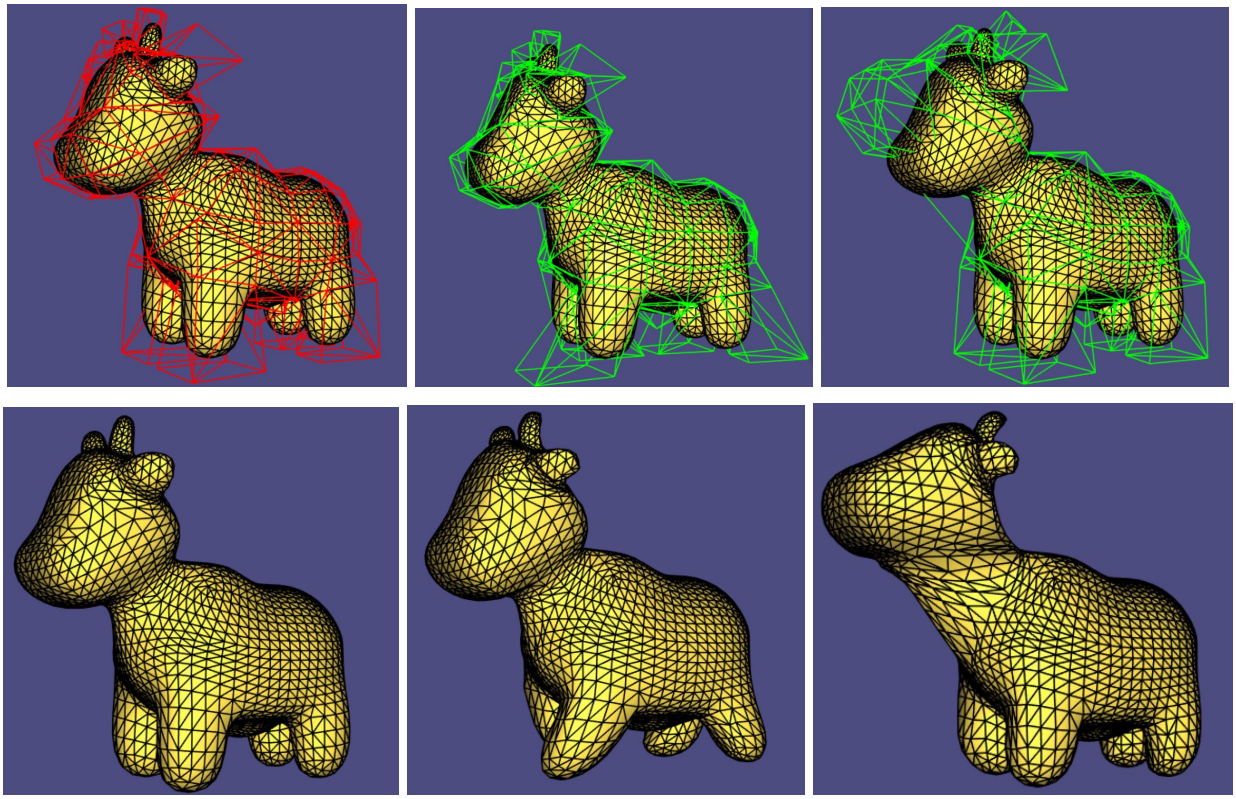

Figure 4. Top Row: Raw Cage; Deformed Cage 1; Deformed Cage 2;
Bottom row: Raw model mesh; Deformed mesh 1; Deformed mesh 2;

The Deformation 1 looks more natural and smooth than the Deformation 2, part of the reason is the control mesh have less control points around the neck area. This is caused by the vertex-based basis coordinate function we used in MVC, as the control cage volume get larger than original cage, the appearance of the result mesh only depends on the weighting of the cage elements the local curvature change of the mesh is not minimised.

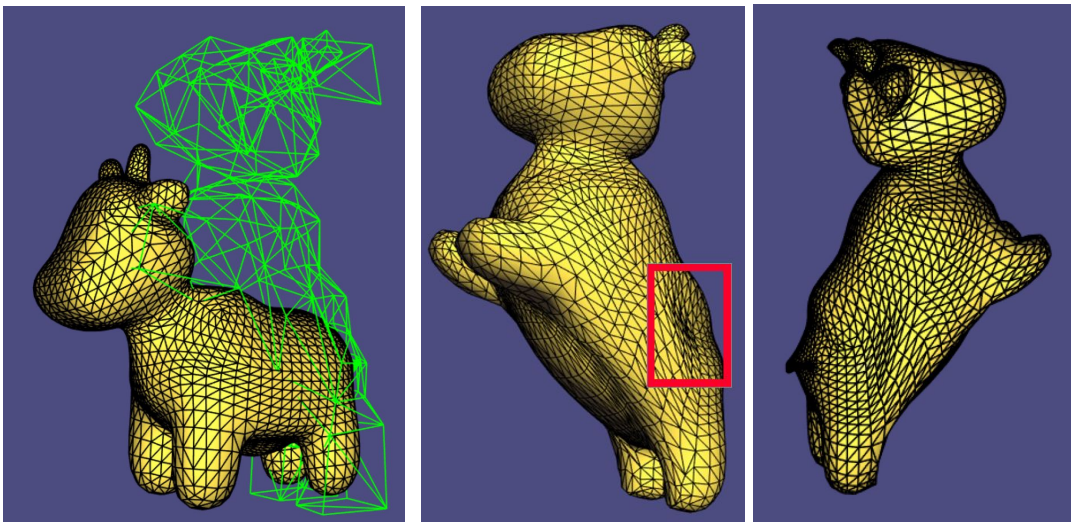

Figure 5: A: Deformation cage 3 ;B: Deformation result 1, redbox marks the artefacts
C: Deformation result 2

Here is a more general and complex deformation of the control cage, you can see the pose of the model mesh fits the target deformation cage well in general, but I've noticed that in some local area (marked in the redbox) appears some artefacts. This area was convex with normal outwards in this area, but after deformation the mesh becomes non-convex and blends inward. I assumed it's due to volumic change around the surrounding area increases lead to this result.

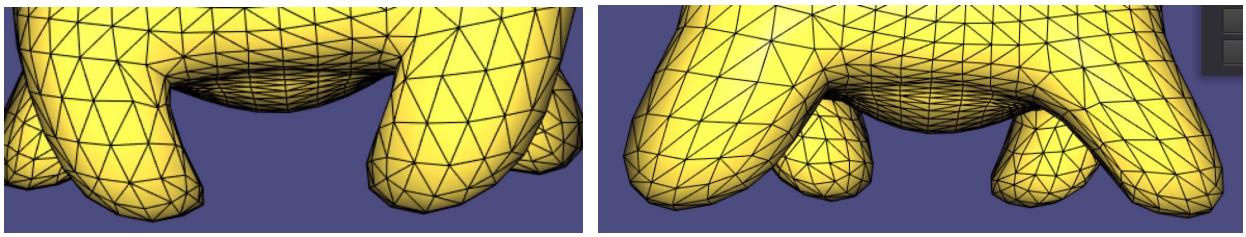

Figure 6: Different effect on walking pose; Bending inwards and bending outwards

I noticed that when blending inwards the curvature of the vertices in the inner side of the mesh is maintained but the outside is stretched. In the other case, when blending outside, the inner side mesh is stretched but the outside curvature is maintained.

One of the property of MVC is that it can interpolates smoothly on both convex and non-convex input, so I have implemented a colour coordinates interpolation to test whether interpolation of cut surface coordinates. The interpolation result supports the paper and the MVC can interpolate non-convex coordinates, smooth interpolation appears in both convex and concave surfaces.

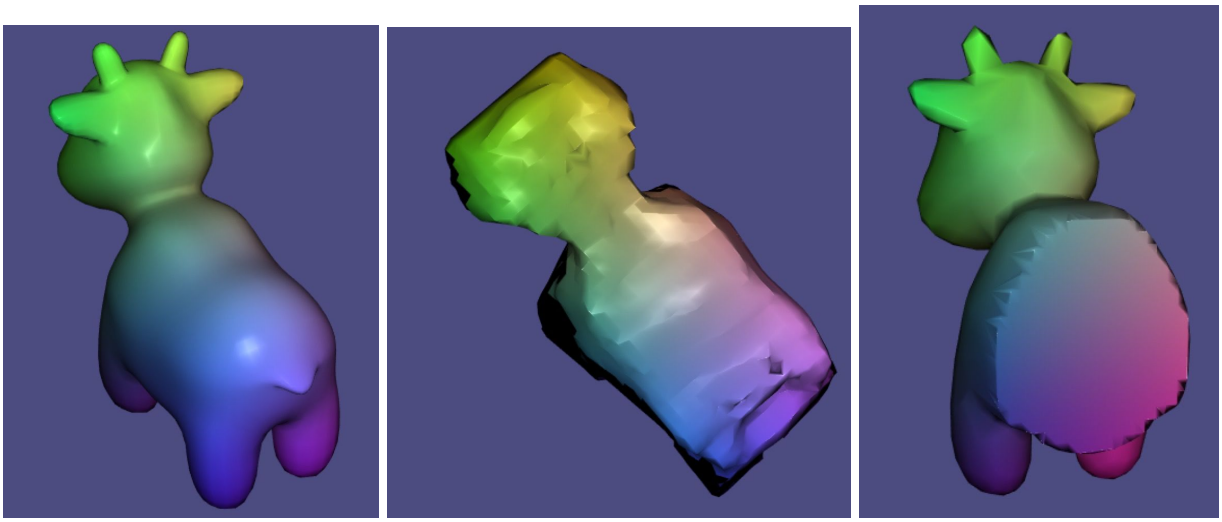

Figure 7. From left to right: a) hue value interpolation of the raw model mesh;
b) hue value interpolation of a horizontal cut, see from above;
c) hue value interpolation of a vertical cut, see from side;

Extension and Novelty

I've choose to construct the control cage of a given mesh as my novelty part of my project without referencing any existing techniques (only based on what we have learned). My aim of this part is not to get a perfectly fit cage that catches every small part of the mesh, as it would get too complicated for some complex mesh like octopus. The aim of this cage is to get a rough estimation of the shape of the cage so user can modify it further in mesh editing softwares.

The main idea is to deform a predefined mesh such as a uniform sphere or cube with different number of vertice by moving vertices in the control mesh towards the closest distance vertex of the model mesh. I have chosen a uniform sphere for this task, any other uniform shapes would do the same job as they are just basically different sampling schemes.

Firstly I calculates the mean centre and bounding box of both mesh, then align the centre of both mesh by translation. The size radius of the sphere mesh must be bigger than the model mesh for the algorithm to run, so I uses a scaling matrix to modify the radius of the sphere mesh. The number of vertices in the control mesh also change the output of my algorithm, ideally we want M>>N, so I've uses a sphere with 252 vertices for testing. The result after alignment is shown below:

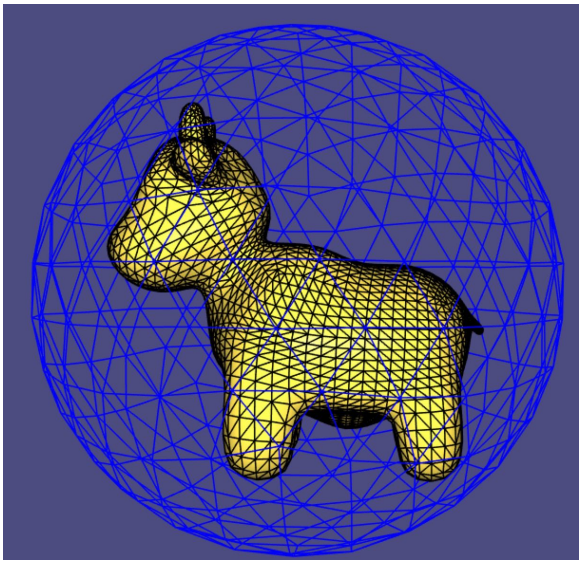

Figure 8. Raw sphere cage after alignment

Then we iterates each vertices in the sphere mesh towards the closest euclidean distance vertex in the model mesh, we only move half length of the total distance each time to avoid overfitting. This step size is programmable, but for now it's set as 0.5* length. We can also calculates a mean direction by including the surrounding vertices in the control mesh as our direction vector. The terminal condition is a defined bias value of distance of the control point with the mesh, as we want the control cage slight larger than the model mesh. This can also be achieved by translating each vertices in the model mesh towards its normal direction to get a scale up version of the model mesh. The figures below show a few iteration of the algorithm:

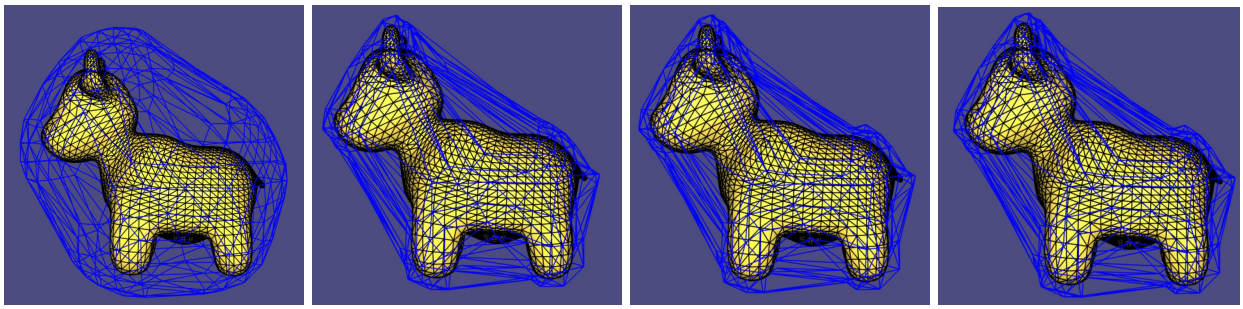

Figure 9. Cage fitting Iteration: 1 itr; 2 itrs; 3 itrs; 5 itrs;

The problem of this approach is that the resulting mesh might be too dense on part of the mesh surface and the result really depending on the initialization of the sphere cage. But this approach guarantees the cage does not intersect with the interior of the model mesh.

I've also implemented another version of this algorithm which applies some jittering to the direction towards the model mesh. This allows the algorithm to find local minimum solution but also introduced some problem of cage vertices intersect with the interior of the mesh and it also converge slower.

The result of including jittering in the cage fitting is shown below:

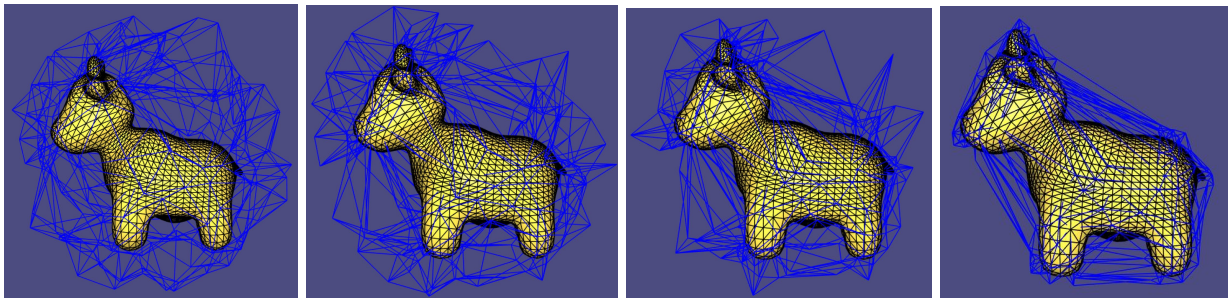

Figure 10. Cage fitting with jittering Iteration: 1 itr; 2 itrs; 4 itrs; 10 itrs;

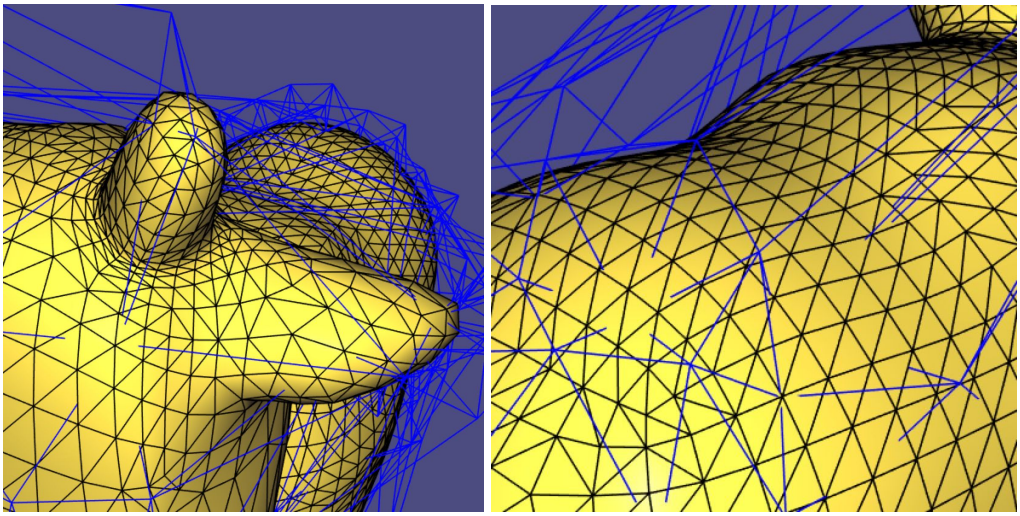

Figure 11. Example of intersection of the interior of the model mesh

Reference:

[1] Barr A H. Global and local deformation of solid primitives. Computer Graphics, 1984, 18(3):21~30

[2] Sederberg y T W, Parry S R. Free-form deformation of solid geometric models. Computer Graphics, 1986, 20(4):151~160

[3] Mean Value Coordinates for Closed Triangular Meshes. Tao Ju, Scott Schaefer, Joe Warren Rice University. Computer Aided Geometric Design Volume 20, Issue 1, March 2003, Pages 19-27