Ritesh Sharma

Homework #4

CS 554

OREGON STATE UNIVERSITY

Term Project Idea

Title: Iso-surface extraction by tessellation of algebraic surfaces using A-Patch Method

Motivation: The main purpose behind this project is to find the mode surface of 3D symmetric tensor field. The mode surface of a tensor is given by a polynomial of degree 6 which is given as,

$$K_1 \, |D|^6 - K_2 \, D^2 = 0$$

Where K_1 and K_2 are constants and $|D|^6$ is frobenius norm and D is determinant of a deviatoric matrix obtained from the decomposition of tensor.

Introduction: Isosurface extraction is one of the commonly used techniques in scientific visualization. Most approaches for isosurface extraction uses variation of marching cubes where the some function are sampled at the grid points. If the signs at the adjacent grid cells differs, then the function passes between the two grid cells and its location can be found with simple root finding algorithms such as bisection or regula falsi method.

These marching cube approaches have difficulties which is shown below:

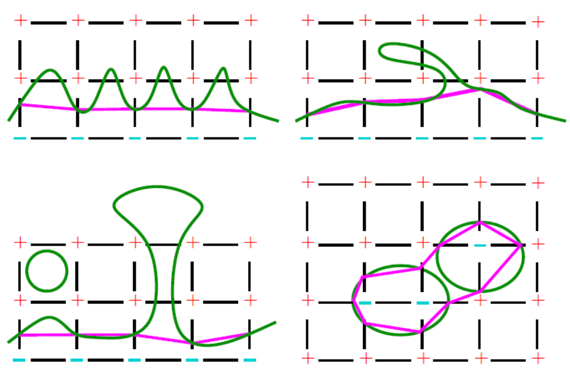


Figure Problem with marching cube algorithm

In the above figure pink curve shows the curve extracted by the marching cube algorithm and green curve is the actual curve which needs to be extracted. Thus we find that the marching cube is not the best approximation for extracting the isocurve/isosurface. Marching cube can easily handle polynomial of degree 2. But in our case, we need to find the surface for the polynomial function of degree 6 which require some modified version of marching cube.

The technical report titled *Using A-Patchs to Tessellate Algebraic Curves and Surfaces* by Stephen Mann proposes novel approach to approximate algebraic curves and surfaces which can handle polynomial function of any degree.

The method proposed in the paper is a form of marching tetrahedron. An A-patch is a Bernstein representation of an algebraic function over a simplex (a triangle or tetrahedron in this paper). The advantage of the A-patch representation is that it guarantees that only a single sheet of the algebraic passes through a triangle/tetrahedral cell, thus avoiding most of the problems mentioned above.

In this paper, author proposes the following algorithm:

Step 1: Convert the cubical grid into five tetrahedrons.

Step 2: Convert algebraic function of each of the tetrahedral into Bernstein representation

Step 3: Check the Bernstein representations in all the five tetrahedrons

- If all five tetrahedrons have strictly positive(negative) coefficients, then the cubical grid does not have any surface passing through it and we do not consider the cubical grid
- If some of the tetrahedron are in A-Patch format and each of the other tetrahedron have coefficients of one sign, then tessellate the A-Patch.
- If any of the Bernstein representations has mixed sign coefficients that are not in A-patch format, then
    - If cube size is above minimum, then discard the Bernstein representations, subdivide the cube into eight sub cubes, and repeat.
    - Else mark cube as unresolved

Once all the A-patches are found, the next step is to perform root finding and then stitch all the surfaces obtained out of each A-Patch to get the isosurface.

It is always important to provide a better and easily accessible GUI for the user to use the software. I have tried to build the GUI as simpler as it can be. Right after the launch of the project, the below screen appears:

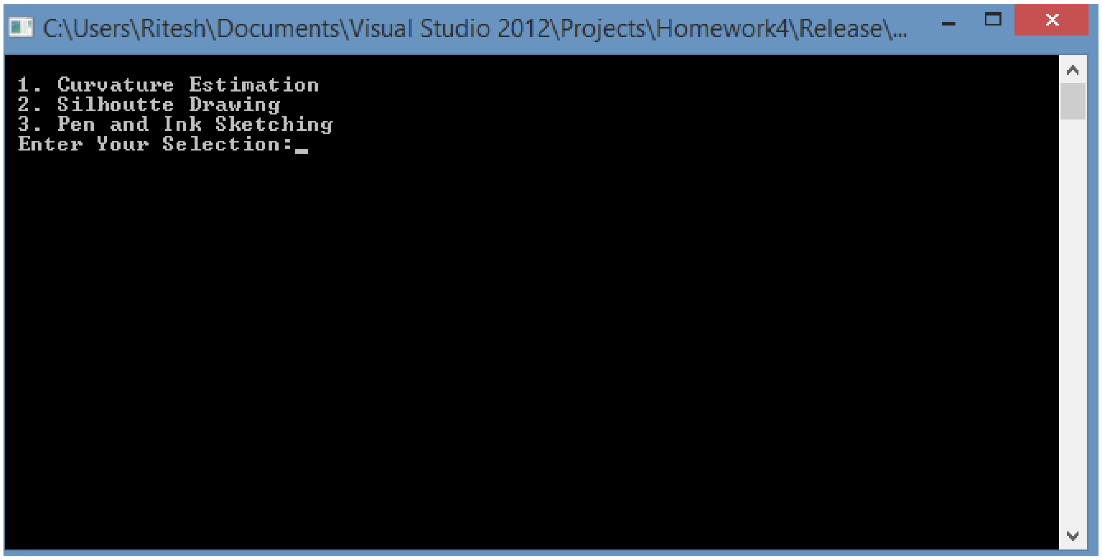


Figure 1. Screen 1

The user can select any option based on his/her choice by entering the number shown against the option. Let user enter 1 and the below screen appears.

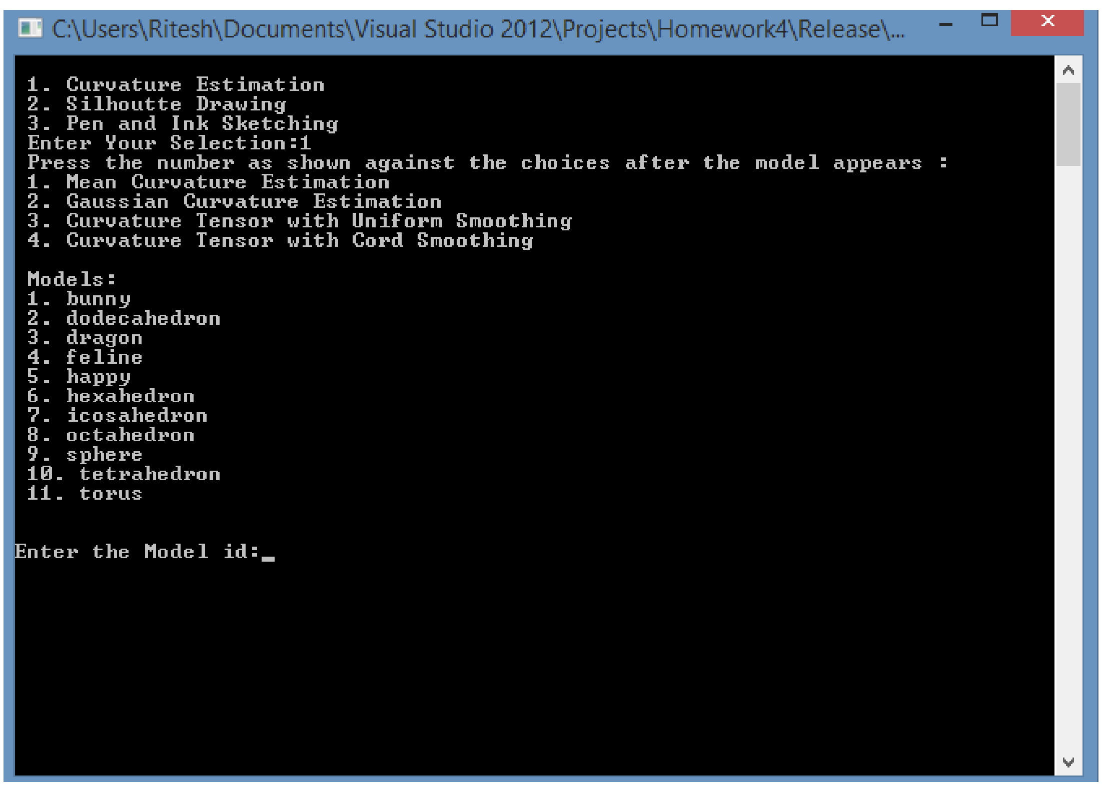


Figure 2. Screen 2

The above screen shows the different choices a user can have after the model is being selected. In the next step, Model id needs to be entered to display the model on the screen. The screen shows wait on the screen and then displays the selected model.

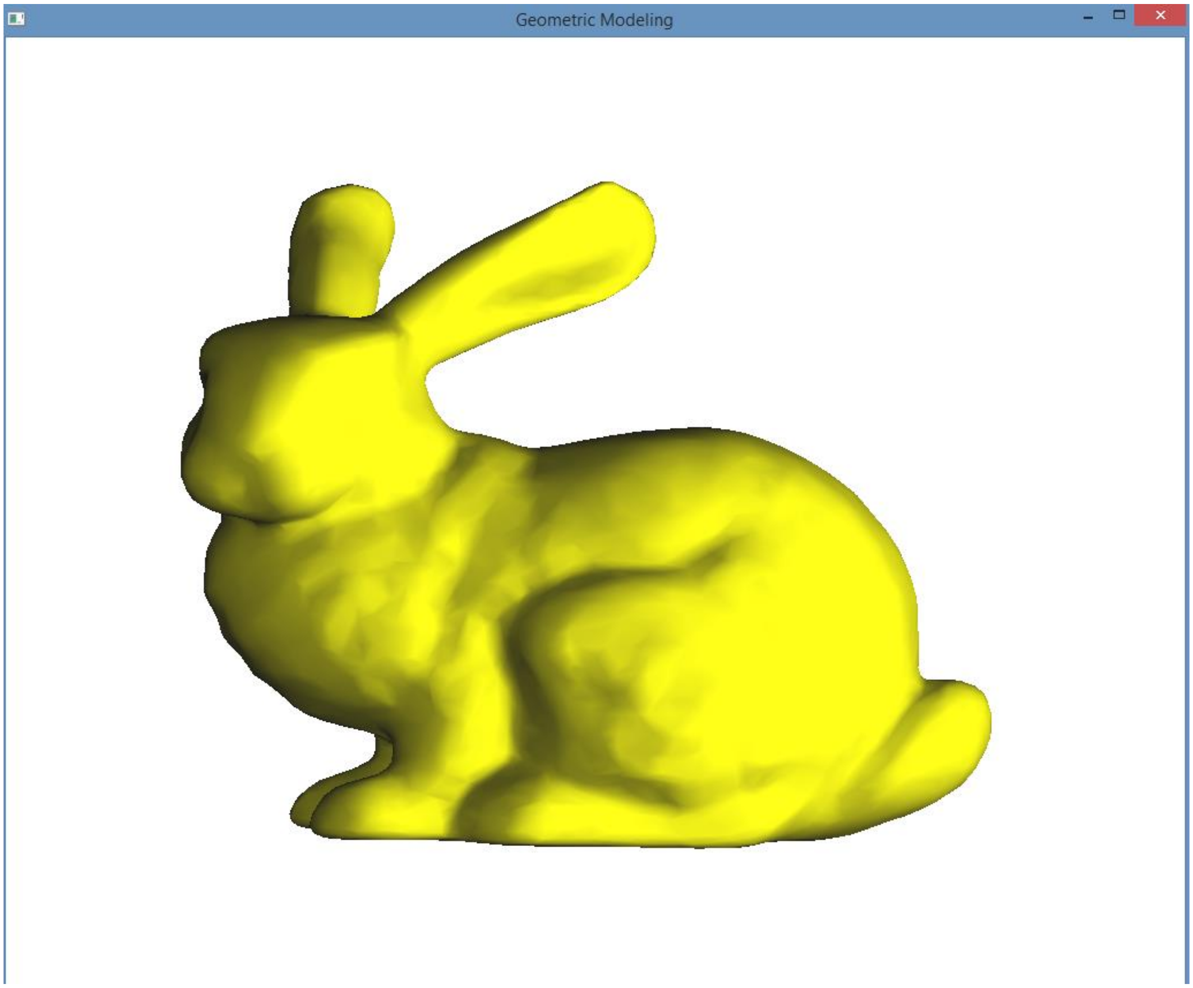


Figure 3. Displaying Model

1. Curvature Estimation

Curvature is an intrinsic property of surfaces. It can be used to identify features such as ridges and valleys, and planar, convex, concave, or saddle shapes. Surfaces can be segmented into regions based on these curvature features, and the segments and features can be used for object recognition and registration algorithms.

The ability to compute curvature from meshes is complicated by the lack of an analytic definition for the surface shape. Meshes are defined at discrete vertices, while curvature is a function of how the surface behaves in a local region around the vertex. It is evident that curvature is based on derivatives and derivatives themselves are a limit function. Thus there is always a noise whenever curvature is estimated over the surface. These noise can be removed using laplacian smoothing. The other factors that need to be considered are the shape and size of the faces of a polygon in a mesh.

In this homework, I will be implementing Mean Curvature and Gaussian Curvature using discrete curvature algorithm described in the paper titled ***Discrete Differential Geometry Operators for Triangulated 2-Maniifolds*** by Meyer et, al.

Mean Curvature and Gaussian Curvature

In the above paper, the mean curvature is computed using a summation to approximate the integral of the Laplacian over the area associated with a vertex which is then normalize by the area associated with that vertex. The area can be a mixtue of Voronoi and Barycentric area, depending on whether or not triangles are obtuse.

The pseudocode (extracted from the above paper) used for calculating mixed area, A_{Mixed} is as follows:

A_{Mixed} =0
For each triangle T from the 1-ring neighborhood of x
    If T is non-obtuse,
        // Add Voronoi formula
        A_{Mixed}+ = Voronoi region of x in T
    Else
        // Add either area(T)/4 or area(T)/2
        If the angle of T at x is obtuse
            A_{Mixed}+ = area(T)/2
        Else
            A_{Mixed}+ = area(T)/4

To compute mean curvature $K(x_i)$:

$$K(x_i) = \frac{1}{2A_{Mixed}} \sum_{j \in N_1(i)} \left(\cot \alpha_{ij} + \cot \beta_{ij}\right) \left(x_i - x_j\right)$$

Where x_i and x_j are the i^{th} and j^{th} vertex, α_{ij} and β_{ij} are the angles opposite to the edge connecting x_i and x_j on the given surface mesh.

Gaussian Curvature is computed using angle deficit method.

To compute Gaussian curvature $K_G(x_i)$:

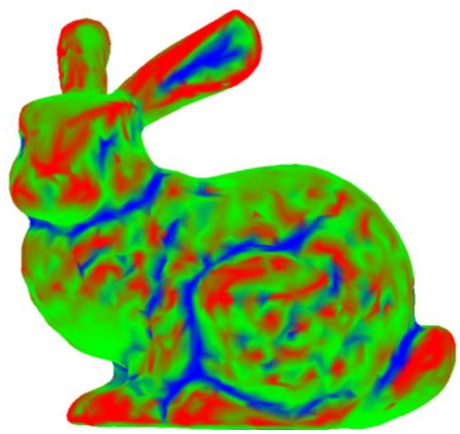
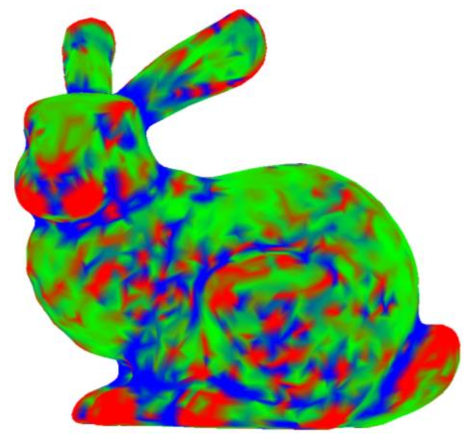
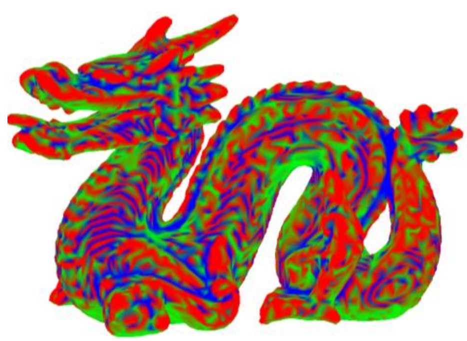
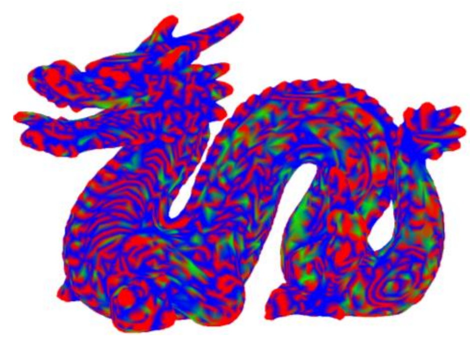
$$K_G(x_i) = \frac{(2\pi - \sum_{j=1}^{\#f} \theta_j)}{A_{mixed}}$$

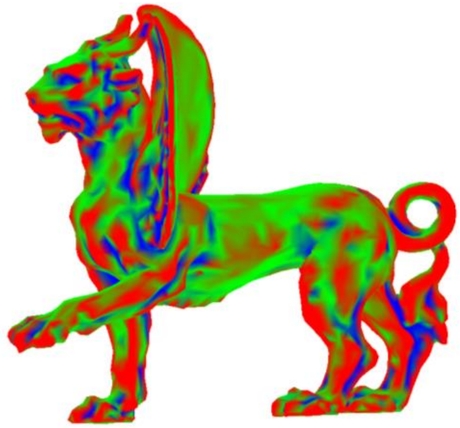
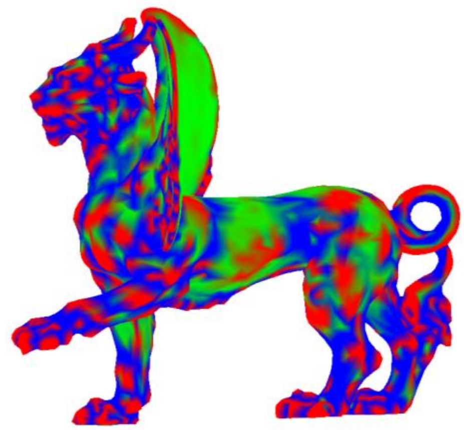
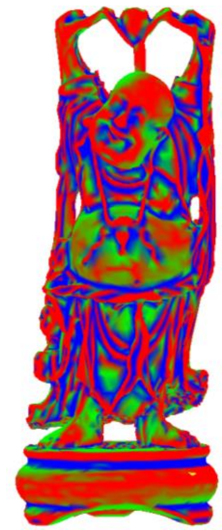
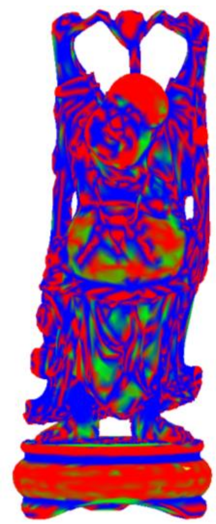
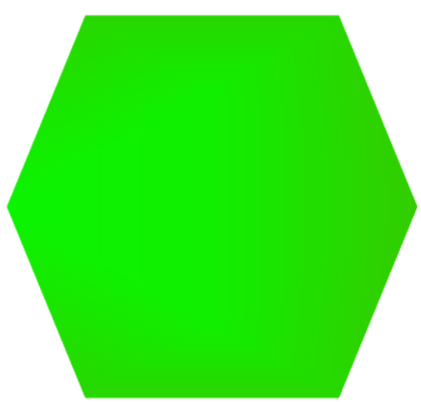
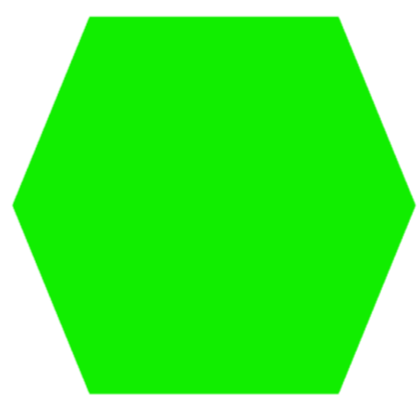
where θ_j is the angle of the j^{th} face at the vertex x_i, and $\# f$ denotes the number of faces around this vertex

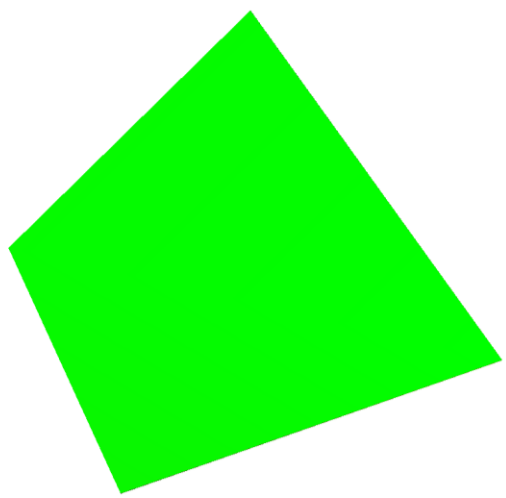
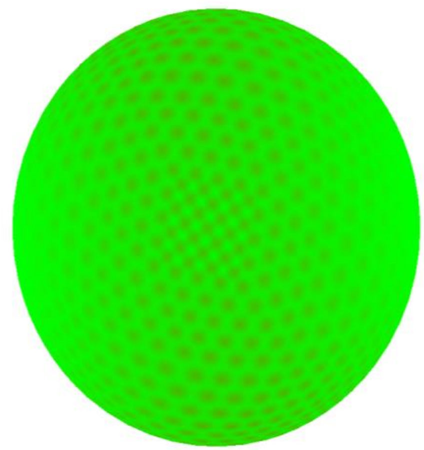
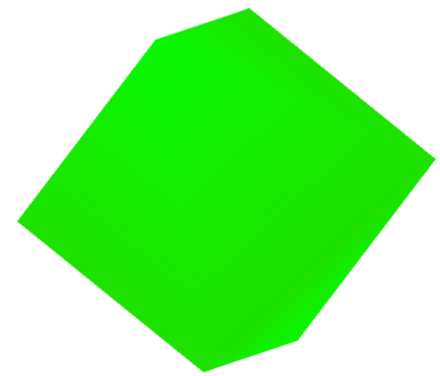
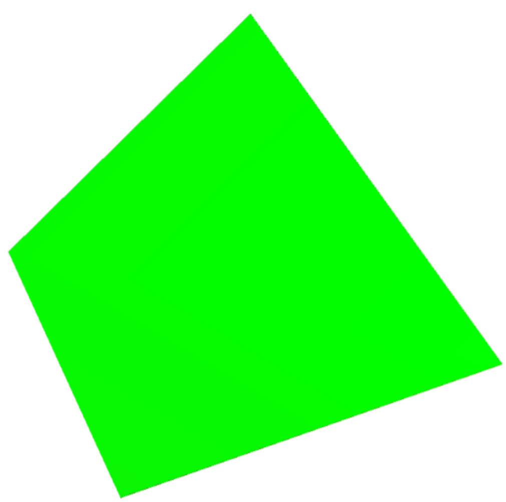
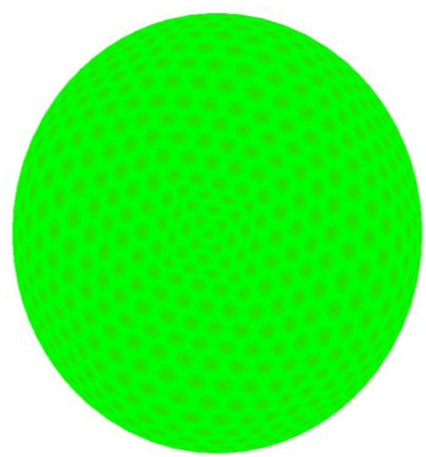
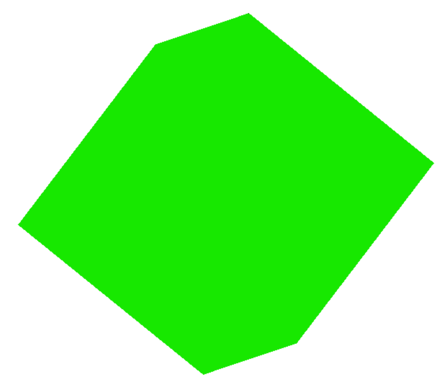
Results

In the results shown below, the red color indicates the positive curvature and blue color indicates negative curvature. Green color represents the curvature where it is close to or equal to zero.

I tried different threshold values to get this easily noticeable visualization which is shown below. For the estimation of mean curvature, -7 and +7 is taken as the threshold value whereas for the estimation of Gaussian curvature, -25 and +25 is taken as the threshold values.

Model	Mean Curvature Estimation	Gaussian Curvature Estimation
Bunny		
Dragon		

Feline		
Happy		
Icosahedron		

Tetrahedron	Sphere	Octahedron
		
		

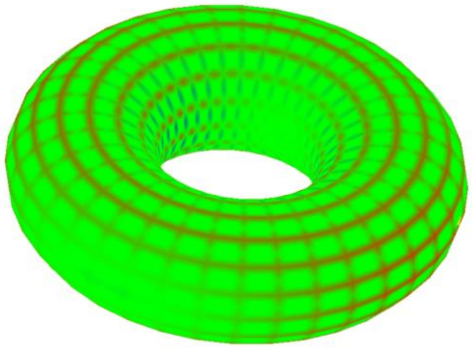
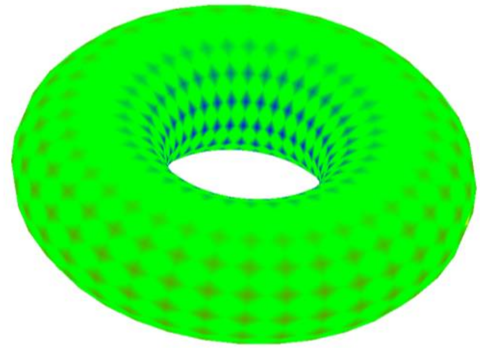
Torus		

Figure 4. Mean and Gaussian Curvature shown over different models

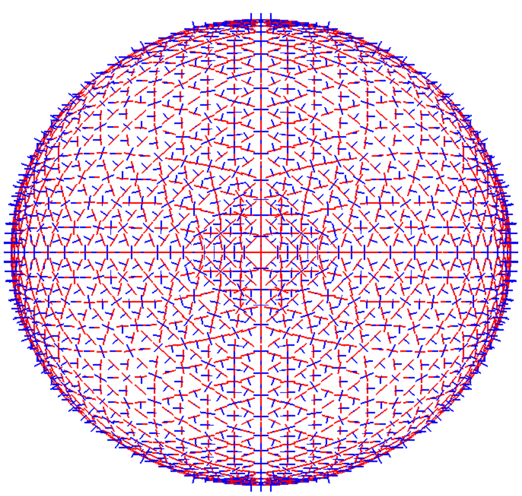
It is observed that the Mean curvature and Gaussian curvature is approximately same for the models such as Tetrahedron, Icosahedron and Octahedron.

Curvature Tensor:

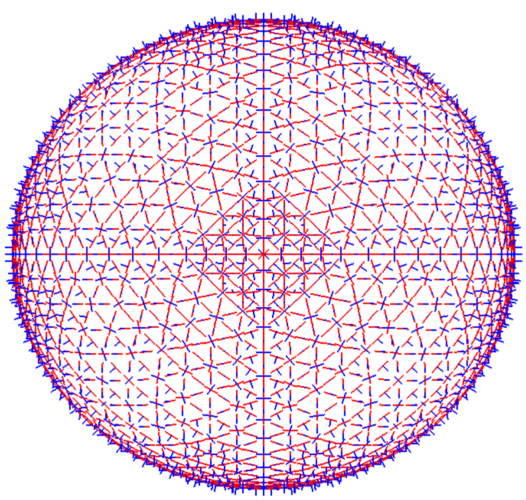
The curvature tensor is shown using cross mark on the vertices of a mesh constituting a model. The red color displays the major eigenvector whereas blue color shows the minor eigenvector.

a.

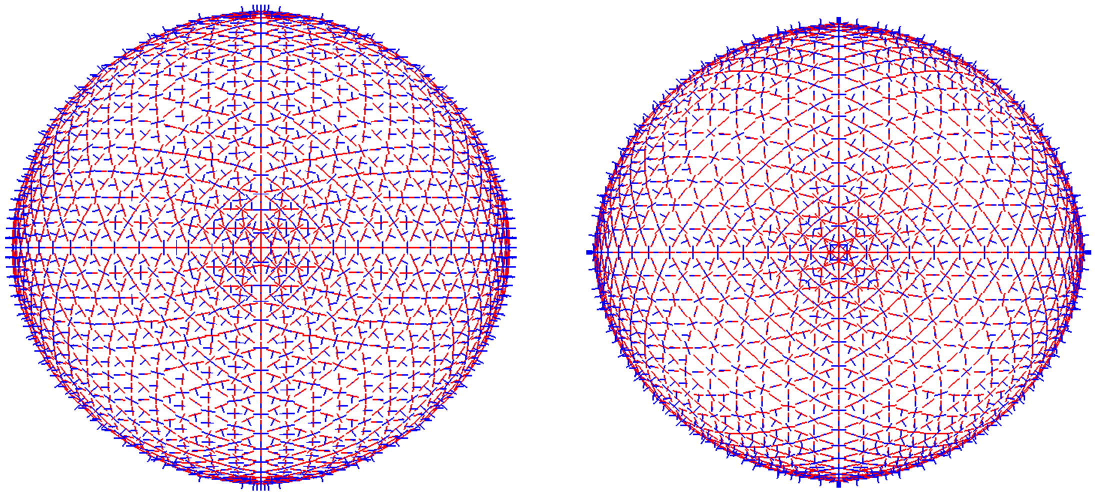
The cord smoothing is found to be more accurate as compared to the uniform smoothing for displaying curvature tensor. It can be seen in the figure 5 below that major eigenvector (in red color) and minor eigenvector (in blue color) are more aligned in cord scheme compared to that in the uniform smoothing. Also cord smoothing converges faster than the uniform smoothing.



without smoothing



without smoothing



after 25 iterations **after 10 iteration**

Figure 5. Curvature tensor after uniform smoothing (left) and after cord smoothing (right)

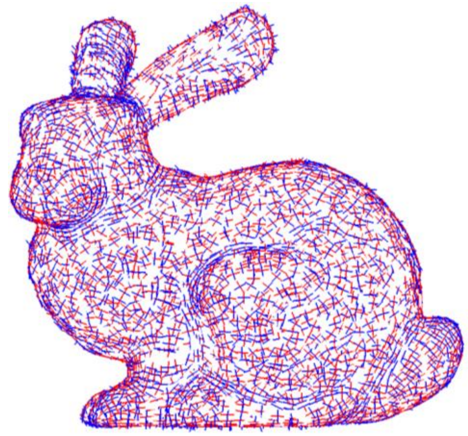
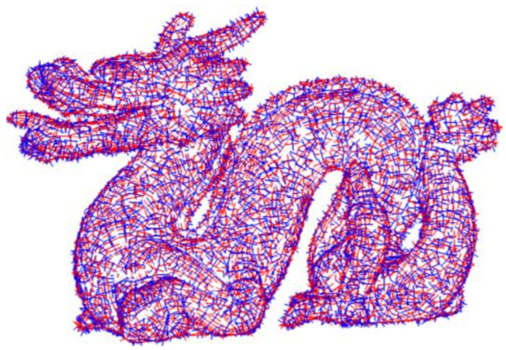
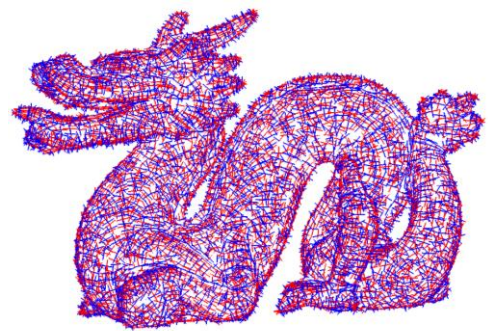
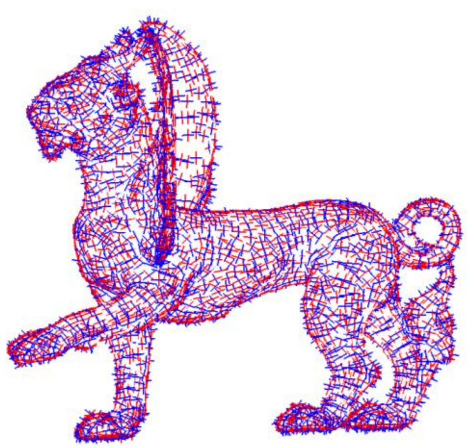
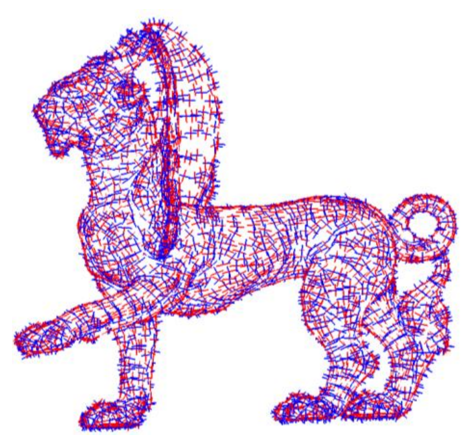
b.

It can be seen in the figure 5 above that with each iteration of smoothing the major and minor eigenvector gets more aligned to each other. The cord scheme makes the changes much faster than the uniform smoothing. I think that since cord scheme uses the edge length as the weights, all the edges becomes equal and thus reduces the curvature with each iteration.

Yes, smoothing improves the visualization of major and minor eigenvector over the model and thus gives us a better estimation of curvature on the model.

I think in case of shapes like tetrahedron, icosahedron, etc., the smoothing will have no changes whatsoever.

Some of the results showing curvature tensor using different smoothing scheme

Model	Curvature Tensor with Uniform Smoothing	Curvature Tensor with Cord Smoothing
Bunny		
Dragon		
Feline		

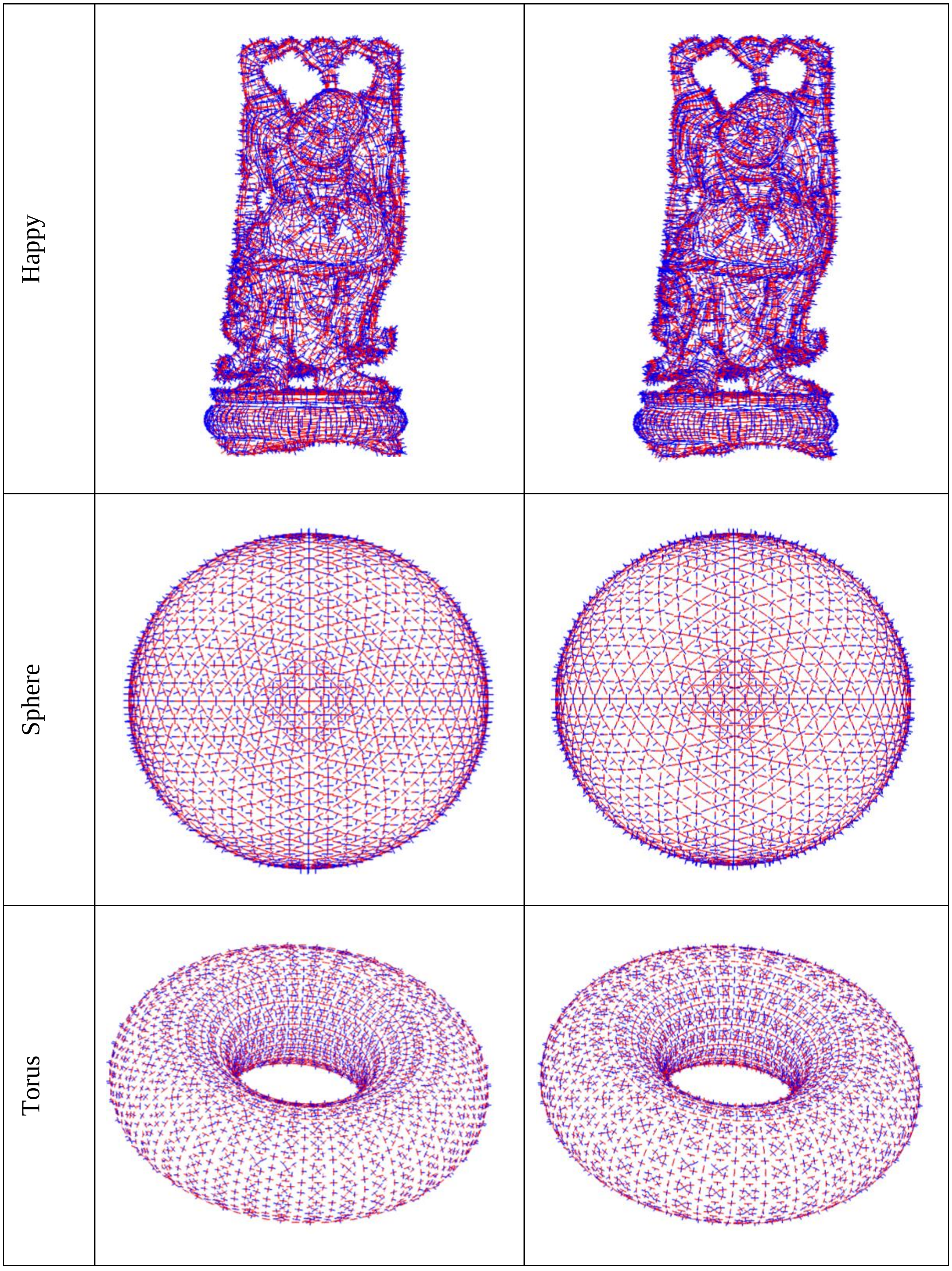


Figure 6. Curvature Tensor shown over different models with different smoothing scheme

2 . Silihouette Drawing:

a.

Figure 7 shows the different view of silhouette drawing based on edges. It can be clearly seen that as the view of the model changes, some part of the model starts disappearing where as some parts starts appearing. As the silhouette drawing is based on the ray-normal dot product test, it is evident that the only the edges of the model which are in front will be visible. In viewpoint 1, some curves on the face on the bunny is visible but when the model is rotated toward left (viewpoint 2) the curves disappears and the torso of the bunny starts showing some extra curves which somewhat gives an idea of curvature over the torso. Similar changes can be seen in the dragon model shown in viewpoint1 and viewpoint 2.

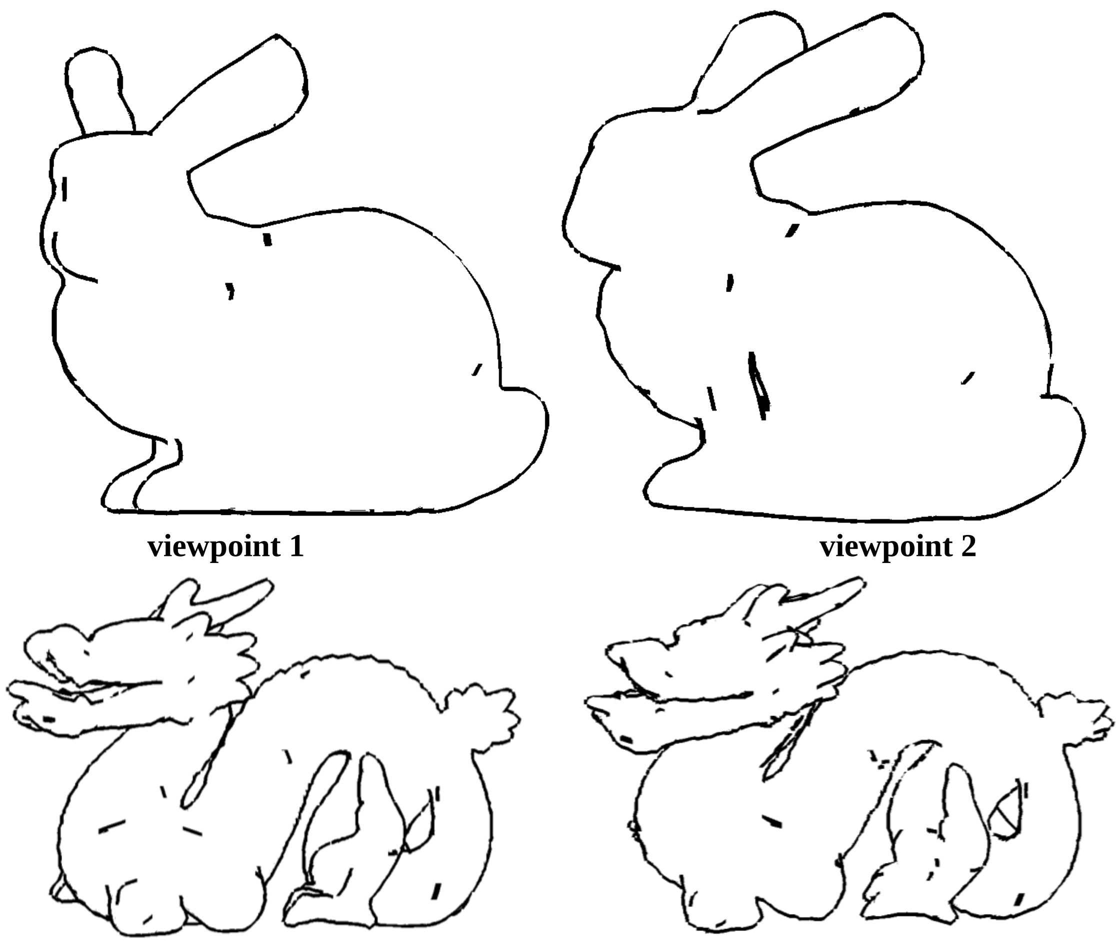


Figure 7. Different view of silhouette drawing based on edges (Method 1)

b.

Figure 8 shows the silhouette drawing based on faces. With change in viewpoint, only the boundary edges of a model is found to change. If bunny and dragon model shown in viewpoint 1 and viewpoint 2 shows slight changes. This method doesn’t reveals any details on the torso of bunny and dragon.

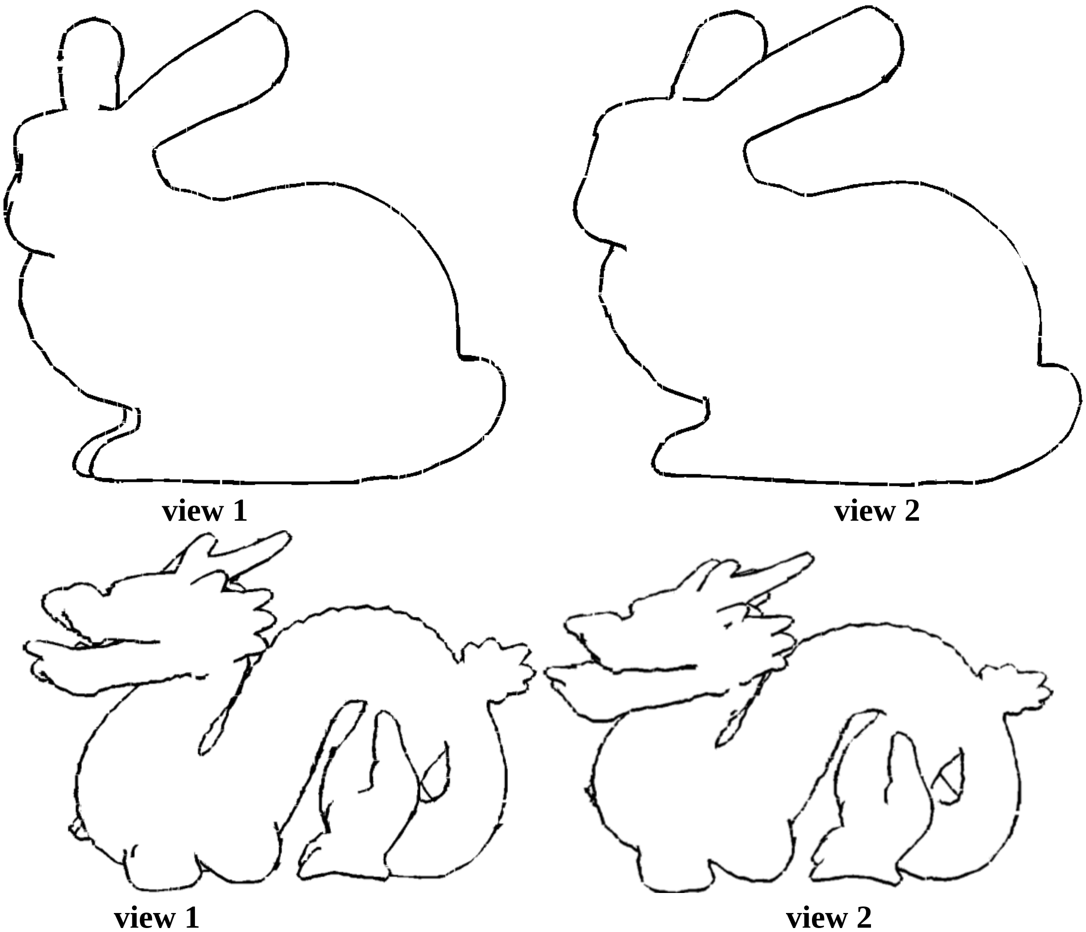
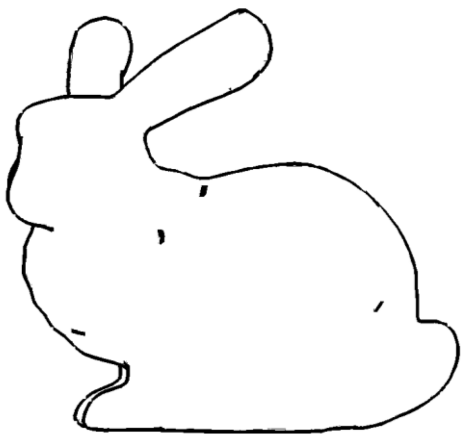
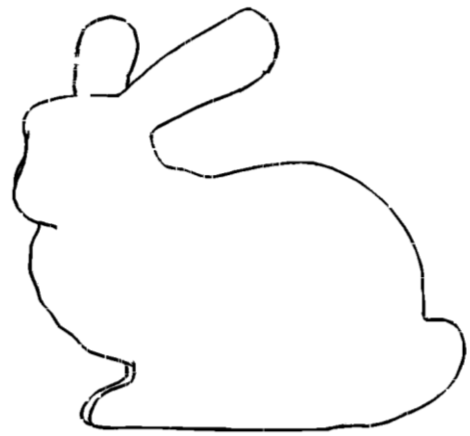
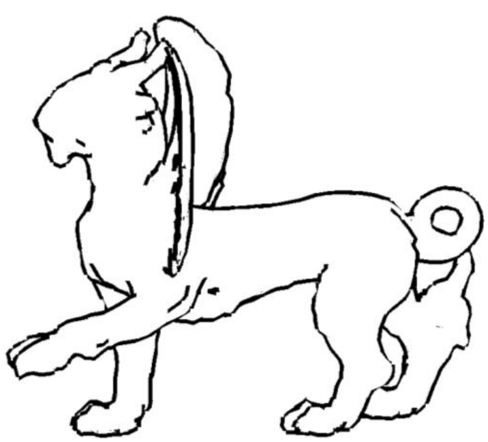
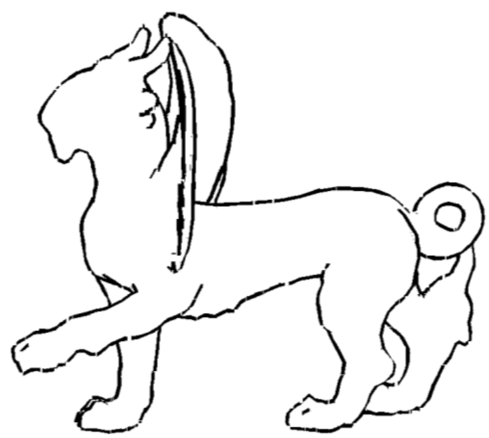


Figure 8. Different view of silhouette drawing based on faces (Method 2)

Method 2 is preferred over method 1 because method 2 can accurately find the significant portion for the silhouette drawing. Method 2 does not changes much with change in viewpoint compared to method 1.

More results:

Model	Silhouette based on edge	Silhouette based on face
Bunny		
Dragon		
Feline		

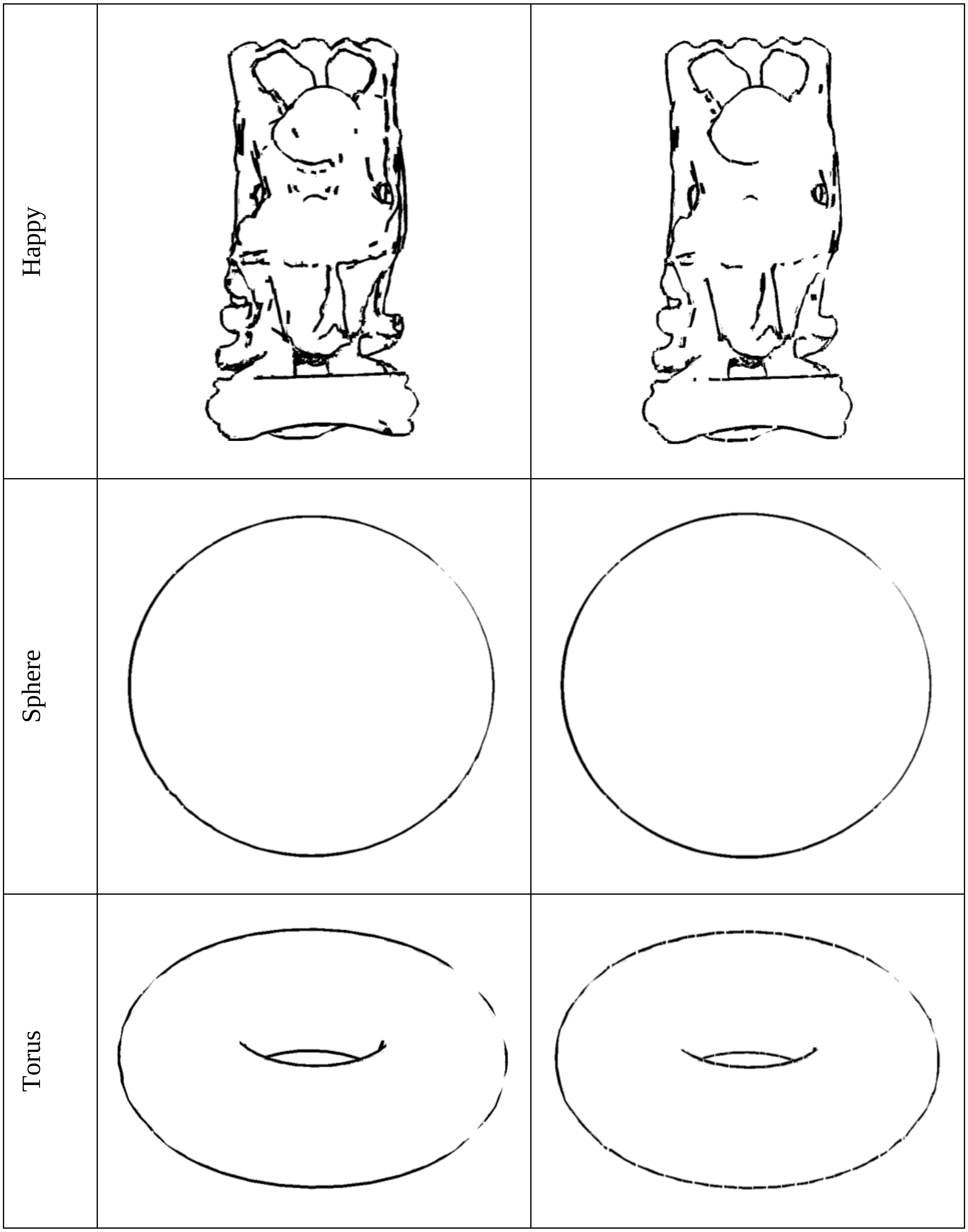
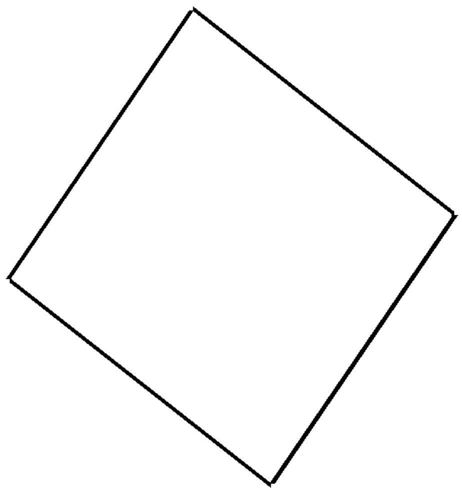


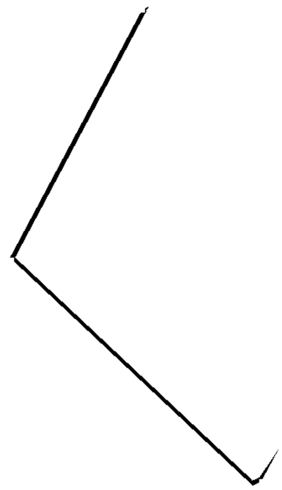
Figure 9. Different view of silhouette drawing based on edges (left) and on faces (right)

c.

From the figure 10 it can be said that neither method 1 nor method 2 captures all the important features of a model. Both method 1 and method 2 have pros and cons.



Method 1



Method 2

Figure 10. Silhouette drawing based on method 1 and method 2

Method 1 captures more information than the method 2 but tends to miss details on the boundary whereas method 2 accurately preserves the details on the boundary. When the viewpoint changes, method 1 starts missing more details than compared to method 2.

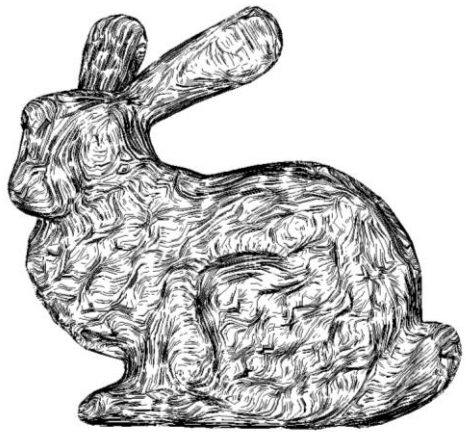
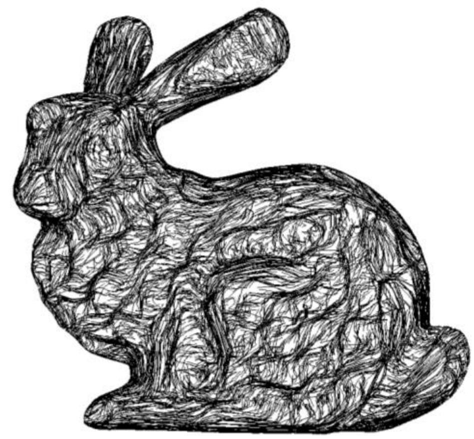
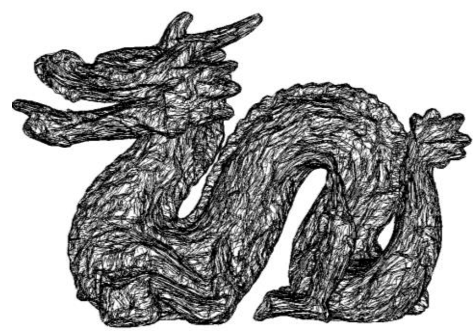
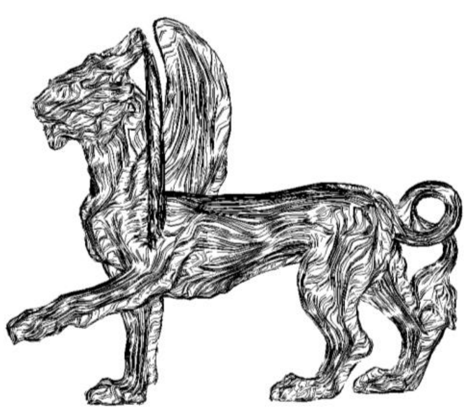
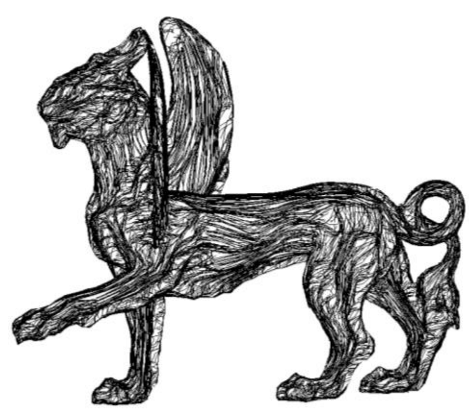
In order to improve the drawing I would like to propose following steps:

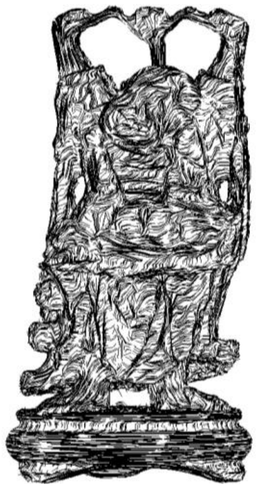
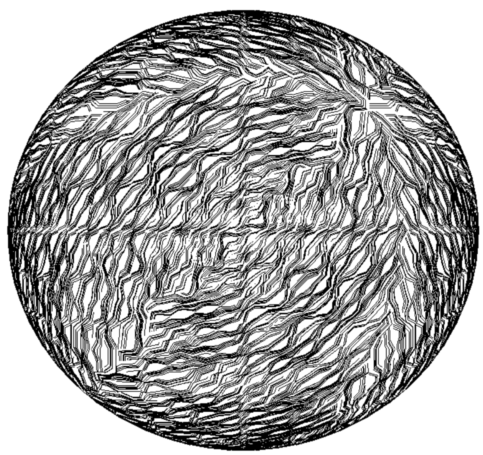
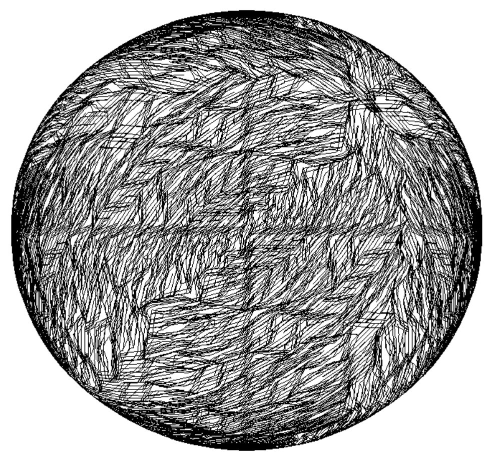
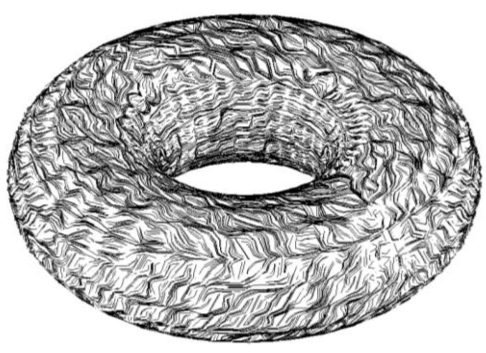
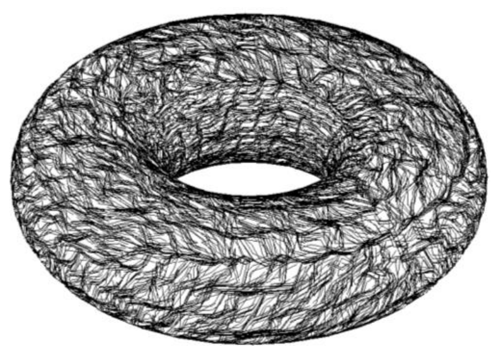
1. Subdividing the mesh in order to have more edges and faces.
2. Tessellation on the portion where the mesh contains a zero level set

3 Pen and Ink Sketching

I have used to method for pen and ink sketching. First method is based on the direction of major eigenvector obtained from curvature tensor. Second method is based on the unfolding of triangle in the same plane and then drawing along the direction of major eigenvector.

The results are shown in figure 11 below:

Model	First Method	Second Method
Bunny		
Dragon		
Feline		

Happy		
Sphere		
Torus		

Yes the two method gives us comparable results. Method 1 gives better results than the method 2. The results obtained from method 1 is superior to method 2 in term of displaying the curvature of the model because method 1 uses principal direction to do the sketching. The curvature in method 2 is not reflected as it should be. In method 2, the triangle in unfolded in the same plane and initial direction is followed when crossing the edges. Thus we don’t get much change in the curvature. Hence the results obtained by method 2 is not as good as method 1.

As method 1 uses principal direction, we may have abrupt changes when crossing from one triangle to other and thus may produce artifacts. In case of method 2, since the initial direction is followed, we have smooth transition from one triangle to other and thus make is somewhat real in appearance.