COMPGV18 Assignment 2 Report

Student Name: Yifei Gao Student ID:13028702

Uniform Laplace:

Mean Curvature

The uniform laplace operator is computationally more efficient as it only requires the connectivity of the mesh, but it’s not a very good estimation of the continuous curvature. It appears to show some non-zero mean curvature at planar surfaces and depends on the quality of triangulation a lot. Figures below shows some result using uniform Laplace operator.

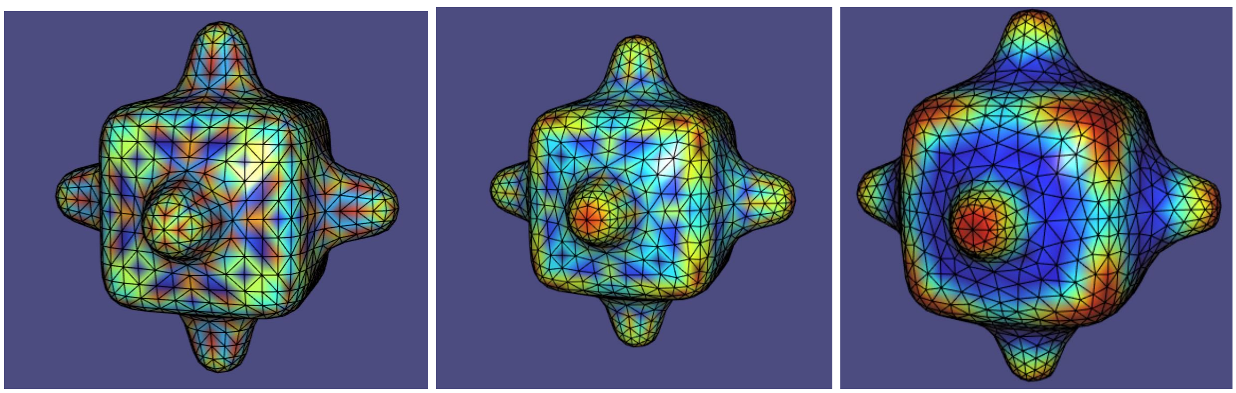

Figure 1. Mean curvature with Uniform Laplace Operator.

This Figure shows how the mean curvature using Laplace operator varies with different triangulation quality, the estimation trends to get better when the quality of triangulation gets better.

Gaussian Curvature

The gaussian curvature is more stable than mean curvature with uniform scheme and it’s obtained by computing the angular deflit around the vertice normalized by surrounding area. The area I used is Barycentric Cells, although its performance is not as good as the other two schemes, but it’s computationally cheaper than others as it does not requires calculation of the Circumcenters. I took ⅓ of the area of faces that is adjacent to the vertices as the normalize area. Figure 2. Shows some result of Gaussian curvature.

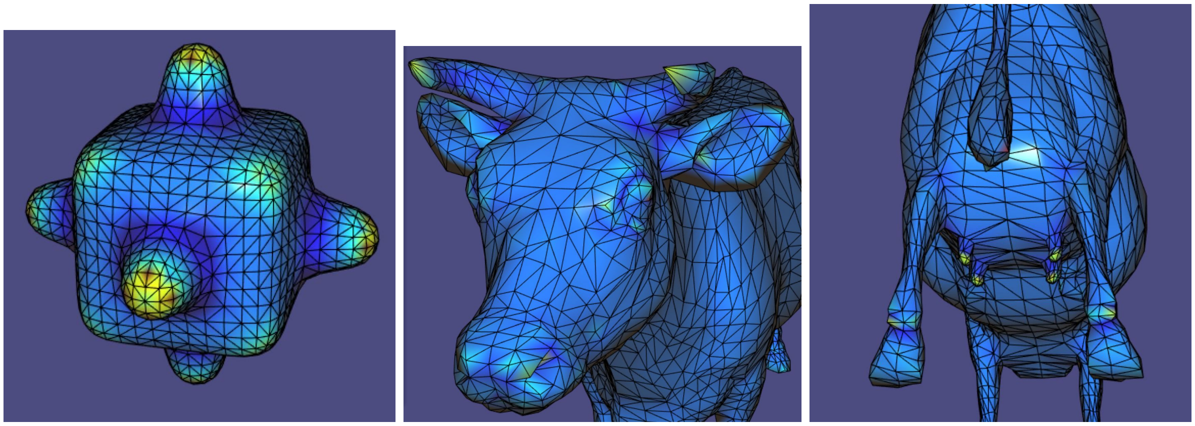

Figure 2. Gaussian curvature results.

The Gaussian curvature shows higher response in the pointing corners of the mesh(i.e the horns of the cow) and low response in the flat area in general.

Non-Uniform(Discrete Laplace):

Mean Curvature

Discrete Laplace-Beltrami shows a more stable and better estimation of the continuous curvature than uniform scheme. As it involves cotangent discretization which trends to be more consistent with the inverse normal direction estimation as the triangulation of the mesh quality is not very good. You can see the improve of curvature estimation of this scheme in the Figure below:

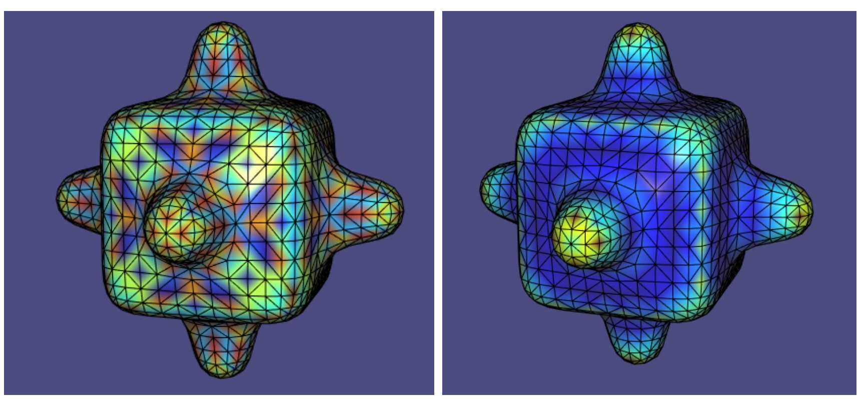

Figure 3. Mean curvature estimation with different scheme
(Lift:Uniform Laplace; Right: Discrete laplace-Beltrami)

For the same mesh we used in uniform scheme, non-uniform scheme shows better consistent on estimating the mean curvature on the planar surface (colored in dark blue) and more evenly distribution of high response in a round conner.

Spectral Analysis

This session is not fully implemented in my code, I have extract the smallest k eigenvectors of the symmetric sparse matrix I've build. But the reconstruction with the Laplace-Beltrami operator does not show the correct result, the possible causes of the error is unidentified and needs further development. Figure below shows the calculated smallest k- eigenvalues and vectors, and the error reconstruction.

```
Eigen values /n
-1.7053e-13
  -0.157268
  -0.157268
  -0.157268
  -0.365947
Eigen vectors /n
  -0.0184256  -0.00784605  -0.00911136    0.0277913  -2.4226e-11
  -0.0328973   -0.0059393  0.000411767    0.0532376    0.0188962
  -0.0328973   0.00694188    0.0243571    0.0472041    0.0188962
```

Figure 4. Eigenvalues and Eigenvector outputs

The resulting 5 smallest eigenvalues I have is size 5 x 1 and the eigenvectors are nx5.

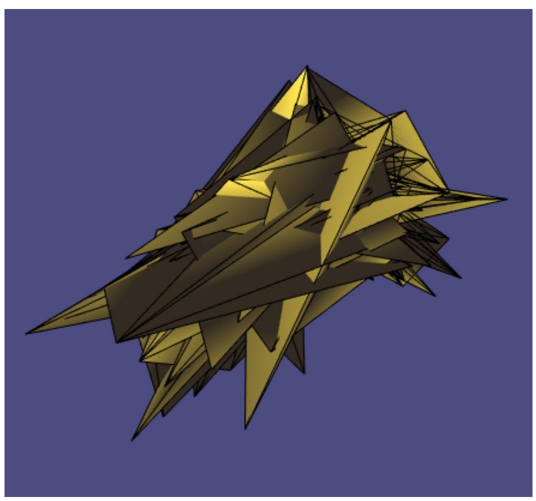

Figure 5. Incorrect reconstruction resulting mesh.

The reconstruction process still need to be debugged, error might comes from the matrix multiplication or smallest eigenvector selection.

Laplacian Mesh Smoothing

Explicit Scheme

The ideal step size for Explicit Scheme is 0.25<**λ**<1.0, for value smaller than this step size, the effect of smooth tooks lots of iteration to become visible and for larger step size, the smooth becomes unstable very quickly. The effect of unstable smoothing with large step size shows the meshes similar to the incorrect reconstruction I did in the spectral analysis. The Figure below shows the smoothing of a simple mesh and a complicated mesh.

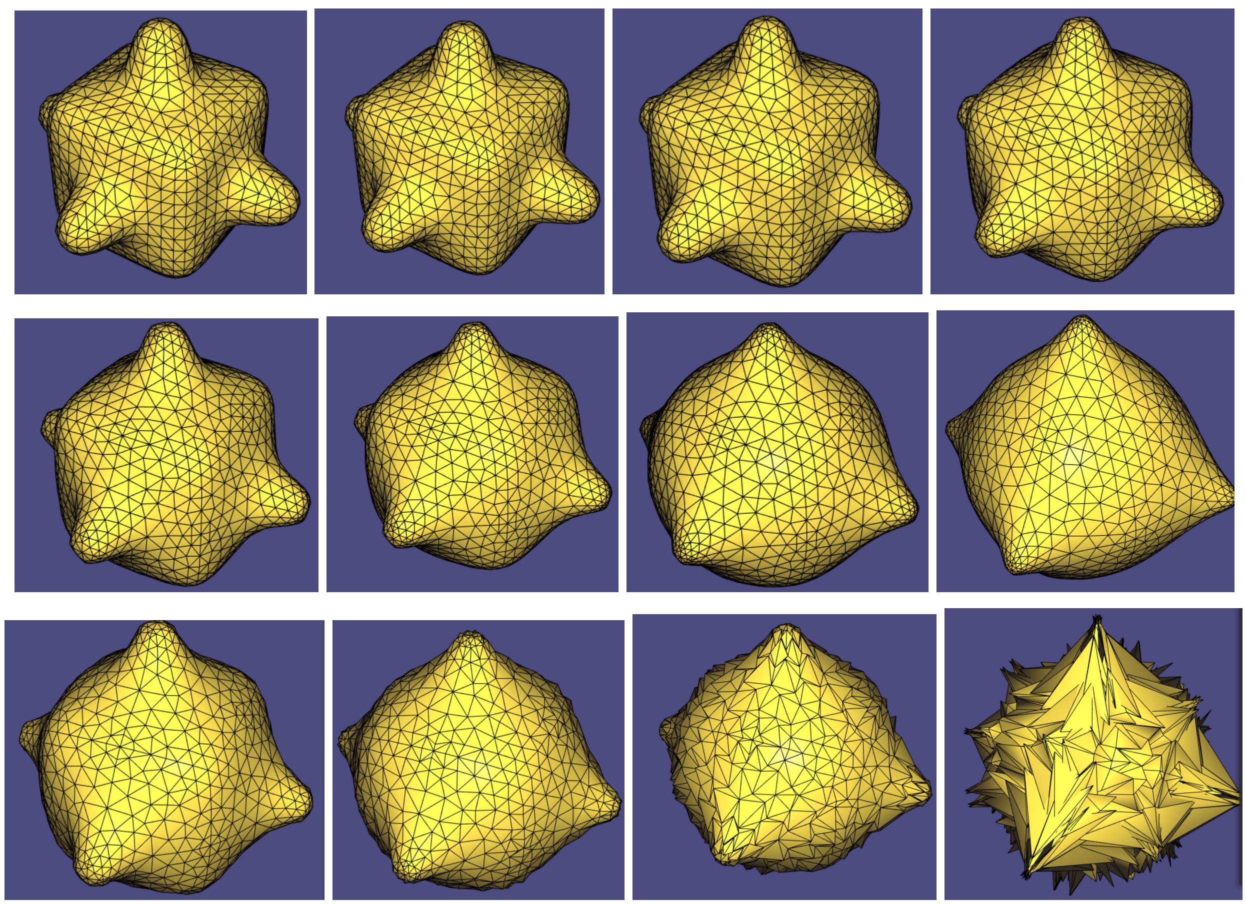

Figure 6.: Explicit smoothing with different **λ** on bumpy.off.
(**λ**=0.1: iterations 5; iterations 10; iterations 15; iterations 30;
λ=0.7: iterations 5; iterations 10; iterations 15; iterations 30;
λ=1.5: iterations 5; iterations 10; iterations 15; iterations 30)

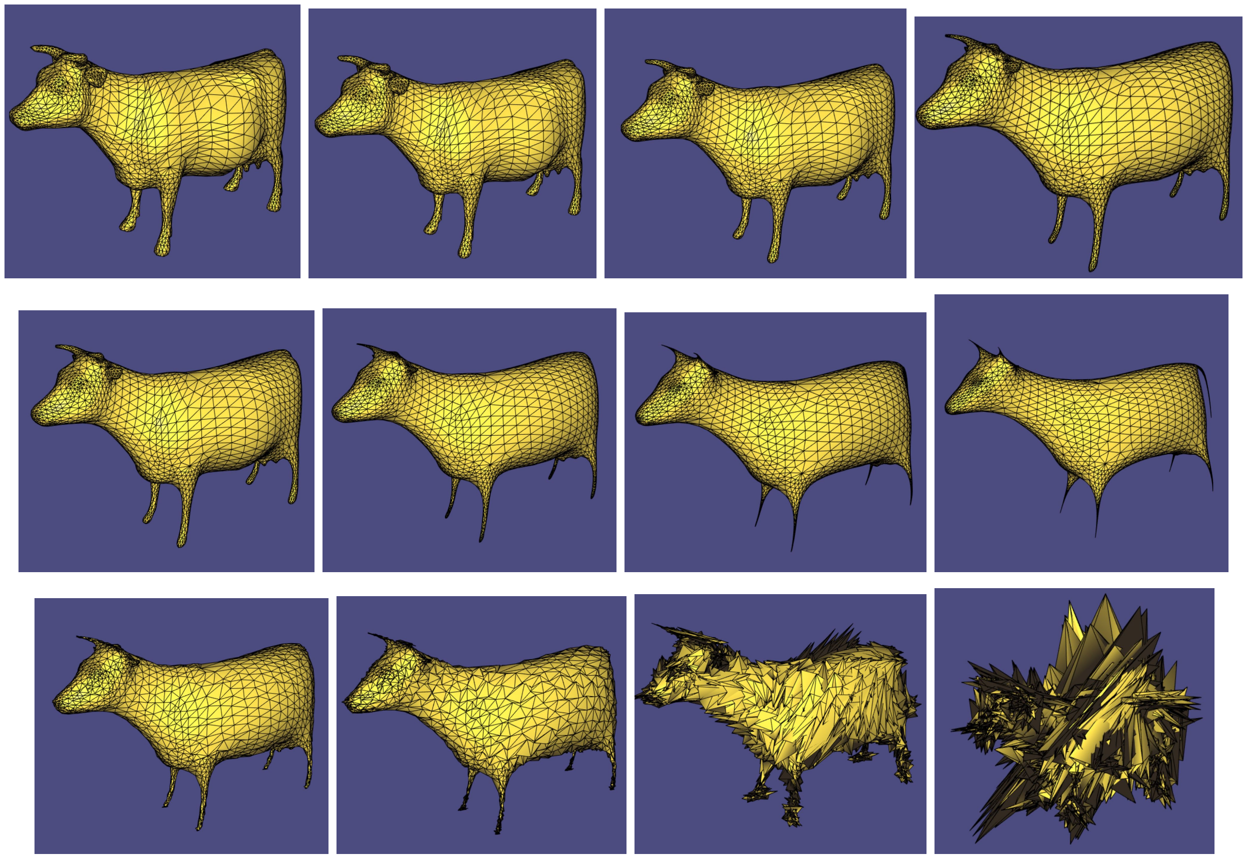

Figure 7.: Explicit smoothing with different **λ** on cow.off.
(**λ**=0.1: iterations 5; iterations 10; iterations 15; iterations 30;
λ=0.7: iterations 5; iterations 10; iterations 15; iterations 30;
λ=1.5: iterations 5; iterations 10; iterations 15; iterations 30)

In general the explicit scheme trends to be very unstable with high **λ** value which lead to overshooting problem in smoothing. But the advantage of the explicit scheme is that it's computational cheaper as it does not require and solving for inverse of the matrices.

Implicit Scheme

In the implicit scheme, I use the uniform-laplacian as L in the implicit scheme, where f at time t+1 is used as reference direction of vertices movement instead of f at time t, as it is in Explicit Scheme. From the result I've tested with increasing **λ** value, the smoothing remains stable even with very large value like **λ**=2.5. The effect of increasing **λ** size is speeding up the convergence of smoothing towards the principal curvature. However, as a pay off to the stability, the computation tooks longer than the explicit scheme as it involves solving inverses of a larger sparse matrix. Figures below shows the implicit smoothing result of the same meshes used in explicit scheme.

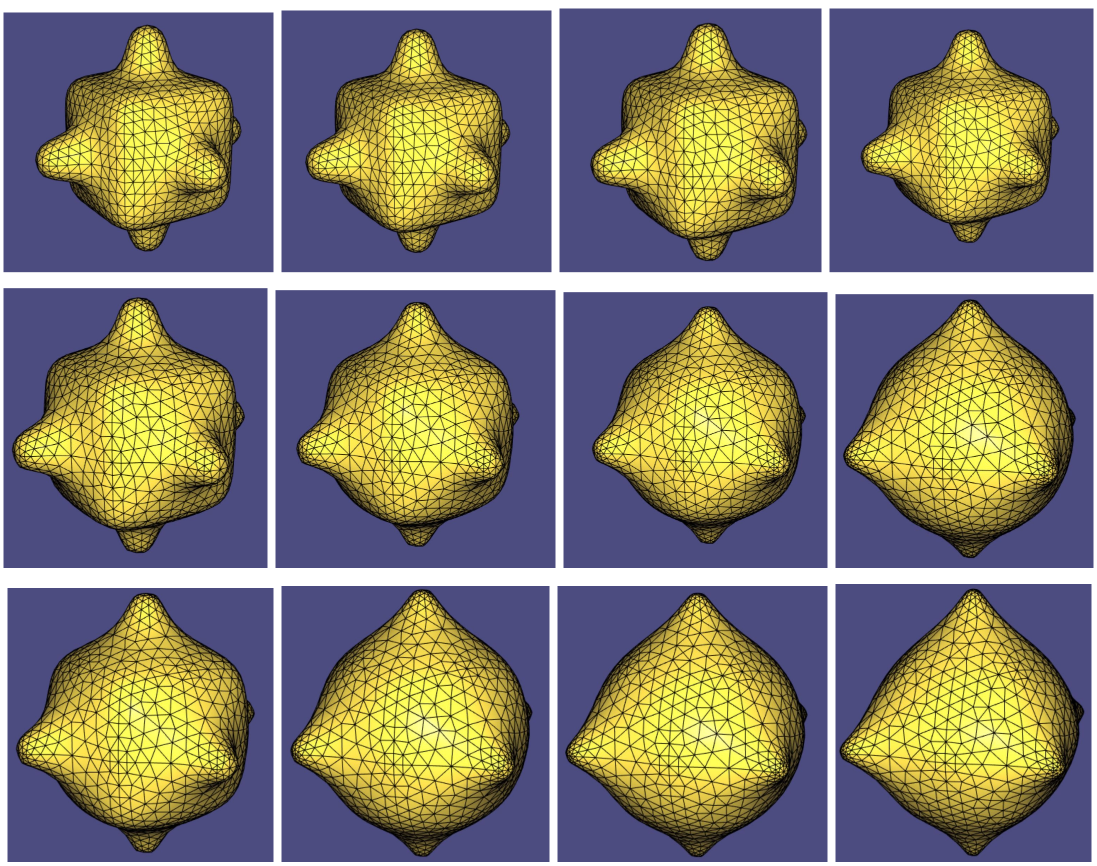

Figure 8.: Implicit smoothing with different **λ** on bumpy.off.
(**λ**=0.1: iterations 5; iterations 10; iterations 15; iterations 30;
λ=0.7: iterations 5; iterations 10; iterations 15; iterations 30;
λ=1.5: iterations 5; iterations 10; iterations 15; iterations 30)

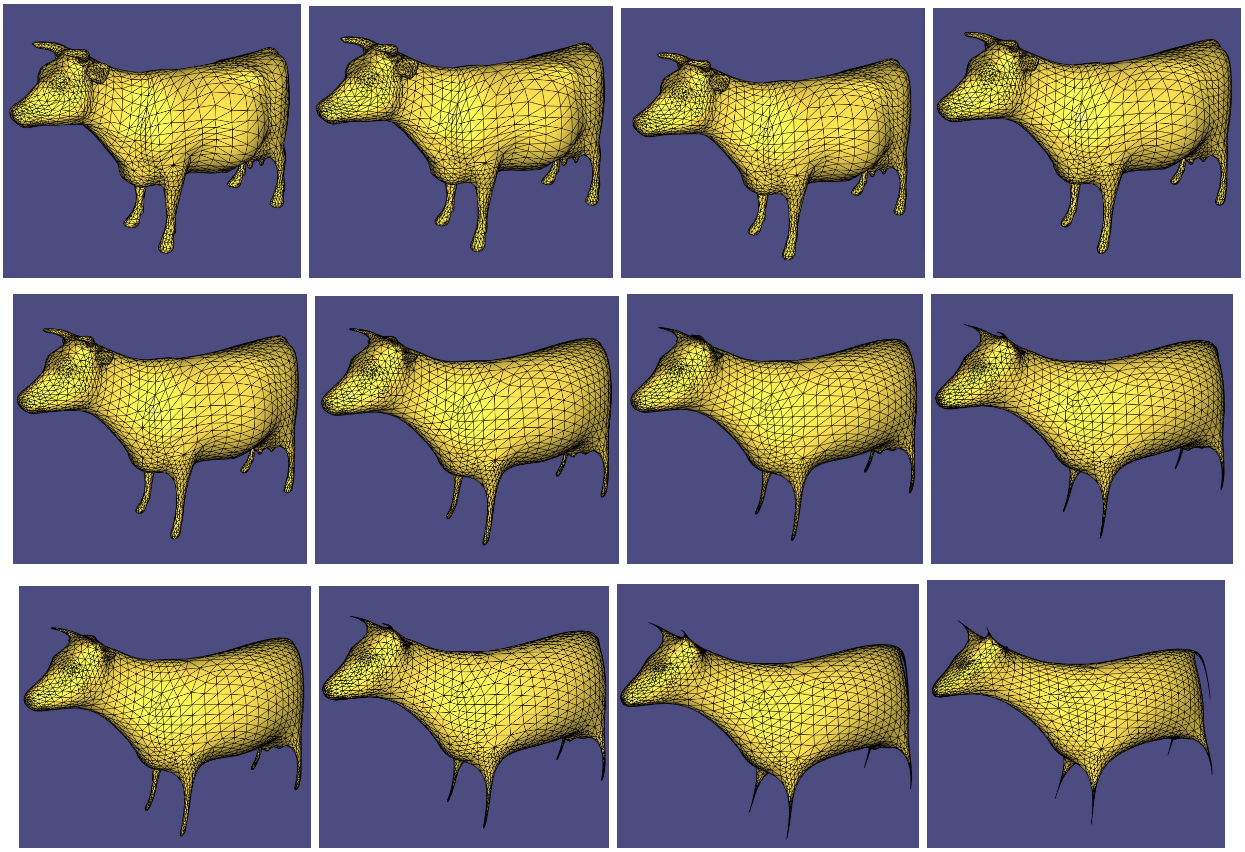

Figure 9.: Implicit smoothing with different λ on cow.off.
(λ=0.1: iterations 5; iterations 10; iterations 15; iterations 30;
λ=0.7: iterations 5; iterations 10; iterations 15; iterations 30;
λ=1.5: iterations 5; iterations 10; iterations 15; iterations 30)

Denosing

I design to test the same mesh with different level of synthetic noise(based on maximum and minimum coefficient of each column), with the same λ=0.5 as step size, to observe the effect and the number of iterations it takes for denoising using the explicit scheme. 'Screwdriver.off' is used as the model for the test. From the test I observed that as the noise level grows, it tooks more iteration to bring the mesh to a surface that seems reasonable smooth, but also as the noise level grows the shape of the object deformed more and more. Some of the deformation caused by the noise is kept in the smoothed result, it trends to be the area where the noises are in the similar direction. Then the denosing starts to fail when the noise level hit a threshold, in my case it is when the noise level >1.5, where the pointing part of the screwdriver starts to become spiral shapes. So my conclusion is that explicit smoothing can reduces noise with a small distance the rest of the point clouds and it works better if the noise distribution is sparse.

The result of different noise level are showed in the figures below:

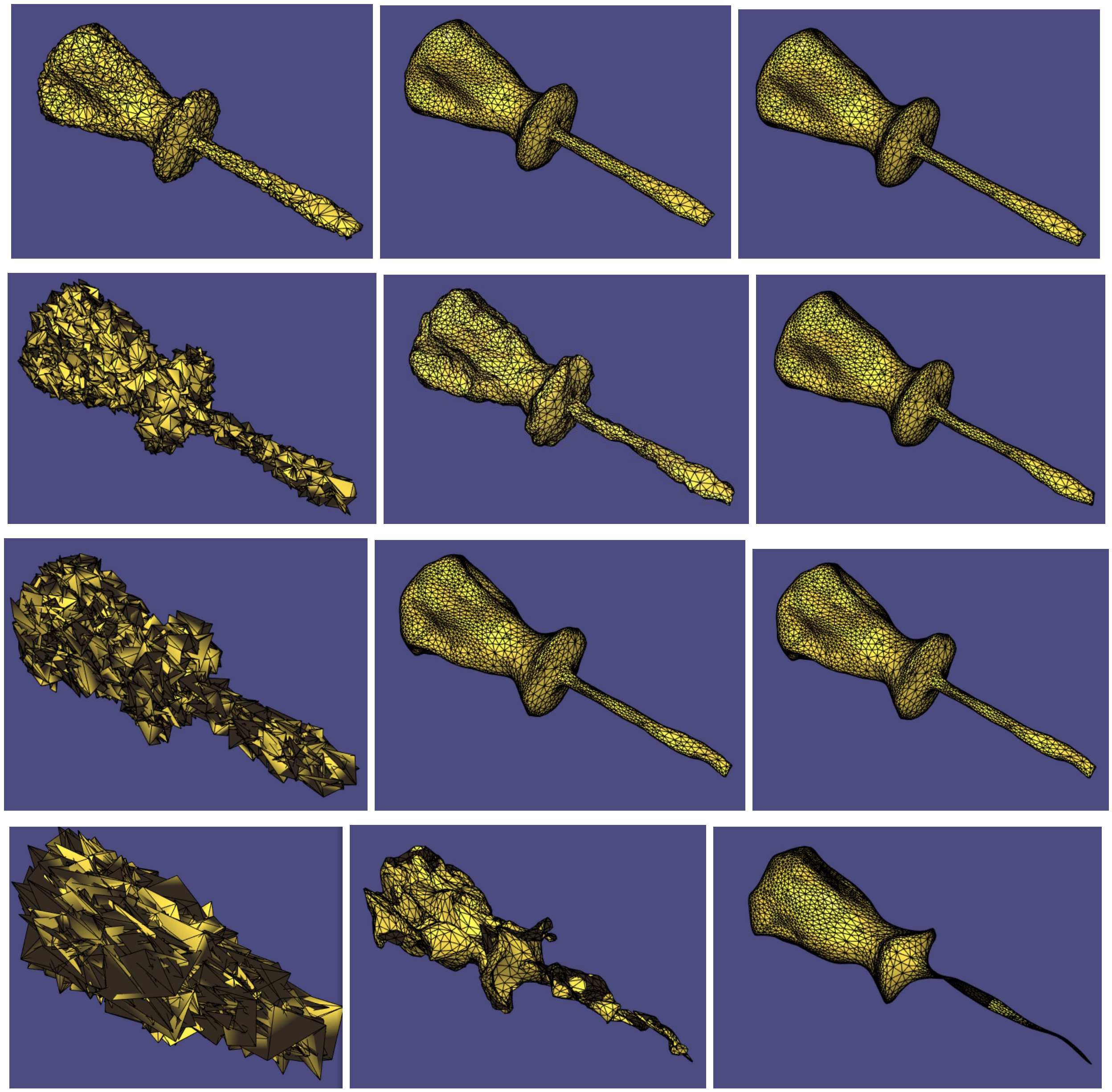

Figure 10.: Denoising result with=0.5 and different noise level on screwdriver.off.
(Noise Level = 0.01: iterations 0; iterations 3; iterations 7;
Noise Level = 0.05: iterations 0; iterations 7; iterations 14;
Noise Level = 0.1: iterations 0; iterations 19; iterations 28;
Noise Level = 0.35: iterations 0; iterations 14; iterations 20;)