Acquisition and Processing of 3D Geometry: Assignment 2

Solutions

Name Yue Wang
Student number 20014458

Question 1 Uniform Laplace:

I choose Barycentric Cells as the form of the area of normalization A_i.

For visualizing mean and Gaussian curvatures separately, the process involves first normalizing the curvature values and then using histogram equalization to enhance the color contrasts. The curvature values are then mapped to colors in the HSV space, with hues representing curvature magnitude. After converting these to RGB, the colors are applied to the mesh vertices. This procedure highlights the curvature variations across the mesh, allowing for an immediate visual assessment of geometric properties.
The regions of high curvature are to be shown in warmer colours (reds and yellows), while areas of lower curvature are in cooler colours (greens and blues).

As Fig. 1 shows, the original mesh is unsmoothed, and the estimation of mean curvature is affected by noise, resulting in inaccurate curvature estimation. In contrast, Gaussian curvature is an intrinsic geometric property of the surface and is unaffected by noise, leading to more accurate curvature estimation.

Question 2 First and Second Fundamental Forms:

To compute the first and second fundamental forms for an ellipsoid given by the parametric equations

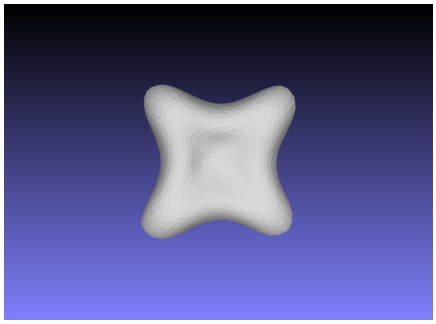
$$p(u, v) = (a\cos(u)\sin(v), b\sin(u)\sin(v), c\cos(v)),$$

where $u \in [0, 2\pi)$ and $v \in [0, \pi]$, we first calculate the partial derivatives with respect to u and v:

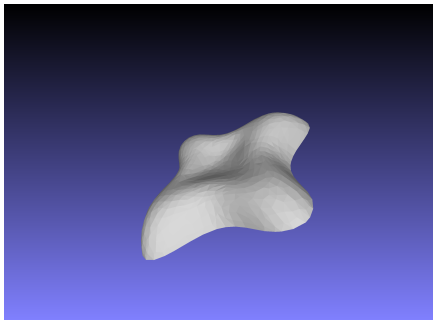
The first derivatives are:

$$p_u = \frac{\partial p}{\partial u} = (-a\sin(u)\sin(v), b\cos(u)\sin(v), 0)$$

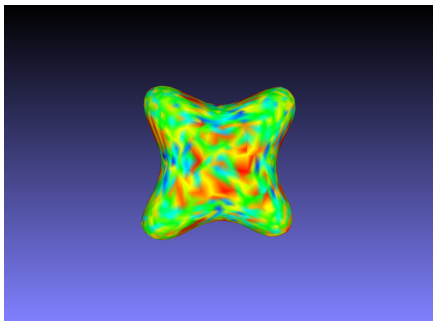
$$p_v = \frac{\partial p}{\partial v} = (a\cos(u)\cos(v), b\sin(u)\cos(v), -c\sin(v))$$



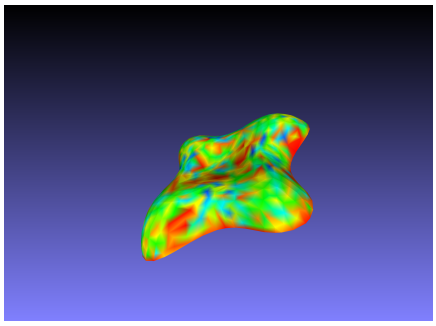
(a) The first viewpoint of the original mesh



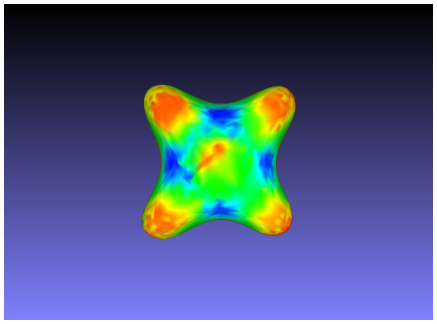
(b) The second viewpoint of the original mesh



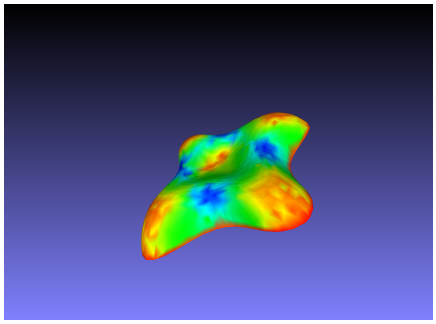
(c) The first viewpoint of the mean curvature



(d) The second viewpoint of the mean curvature



(e) The first viewpoint of the Gaussian curvature



(f) The second viewpoint of Gaussian curvature

Figure 1: Visualization results.

The second derivatives are:

$$p_{uu} = \frac{\partial^2 p}{\partial u^2} = (-a\cos(u)\sin(v), -b\sin(u)\sin(v), 0)$$

$$p_{uv} = \frac{\partial^2 p}{\partial u \partial v} = (-a\sin(u)\cos(v), b\cos(u)\cos(v), 0)$$

$$p_{vv} = \frac{\partial^2 p}{\partial v^2} = (-a\cos(u)\sin(v), -b\sin(u)\sin(v), -c\cos(v))$$

The normal vector N can be found using the cross product of p_u and p_v:

$$N = \frac{p_u \times p_v}{\|p_u \times p_v\|}$$

The first fundamental form (I) coefficients E, F, and G are given by:

$$E = p_u \cdot p_u$$

$$F = p_u \cdot p_v$$

$$G = p_v \cdot p_v$$

The second fundamental form (II) coefficients L, M, and N are given by:

$$L = p_{uu} \cdot N$$

$$M = p_{uv} \cdot N$$

$$N = p_{vv} \cdot N$$

Given a direction vector d on the tangent plane at $p(u, v)$, the normal curvature κ_n in that direction is given by:

$$\kappa_n = \frac{Ldu^2 + 2Mdudv + Ndv^2}{Edu^2 + 2Fdudv + Gdv^2}$$

At the point $(a, 0, 0)$, corresponding to $u = 0$ and $v = \frac{\pi}{2}$, we can evaluate these forms and find the normal curvature κ_n for a direction vector d on the tangent plane.

With these evaluated forms at $u = 0, v = \frac{\pi}{2}$, we have:

$$E = b^2, \quad F = 0, \quad G = c^2$$

$$L = 0, \quad M = 0, \quad N = -c$$

Given a direction vector d on the tangent plane at $p(u, v)$, represented by its components (du, dv), the normal curvature κ_n in the direction of d is given by the formula:

$$\kappa_n = \frac{Ldu^2 + 2Mdudv + Ndv^2}{Edu^2 + 2Fdudv + Gdv^2}$$

At the point $(a, 0, 0)$, if the direction vector d is aligned along the y-axis, which is the major axis of the ellipse section in the yz-plane, then $du = 0$ and $dv = 1$. This simplifies the normal curvature equation to:

$$\kappa_n = \frac{Ndv^2}{Gdv^2} = \frac{N}{G}$$

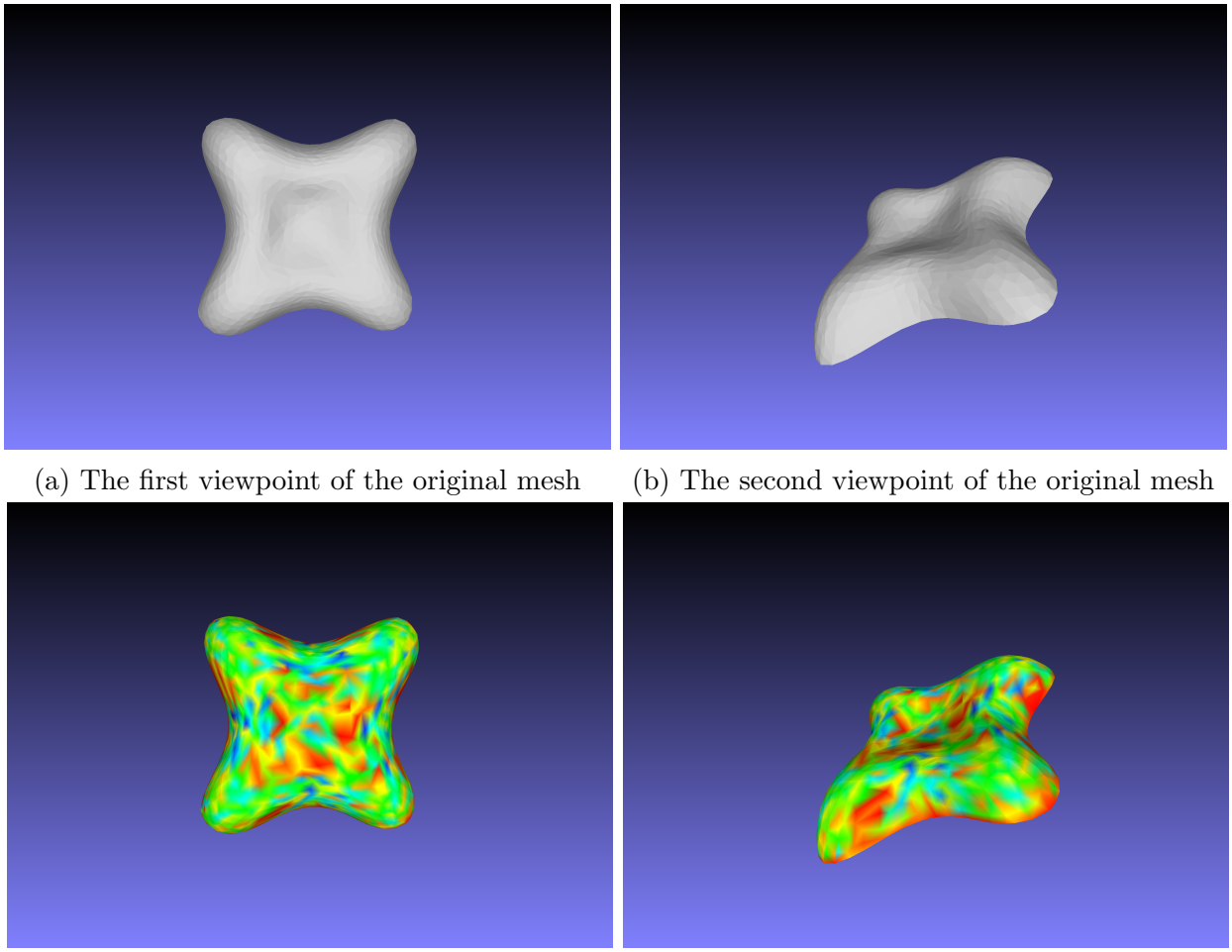
Since $L = 0$ and $M = 0$, and considering our direction vector, the normal curvature κ_n simplifies further to:

$$\kappa_n = \frac{-c}{c^2} = \frac{-1}{c}$$

This result tells us that the normal curvature at the point $(a, 0, 0)$ in the direction of the y-axis is the reciprocal of the parameter c of the ellipsoid, with a negative sign indicating the direction of curvature towards the center of the ellipsoid along the y-axis.

Question 3 Non-uniform (Discrete Laplace-Beltrami):

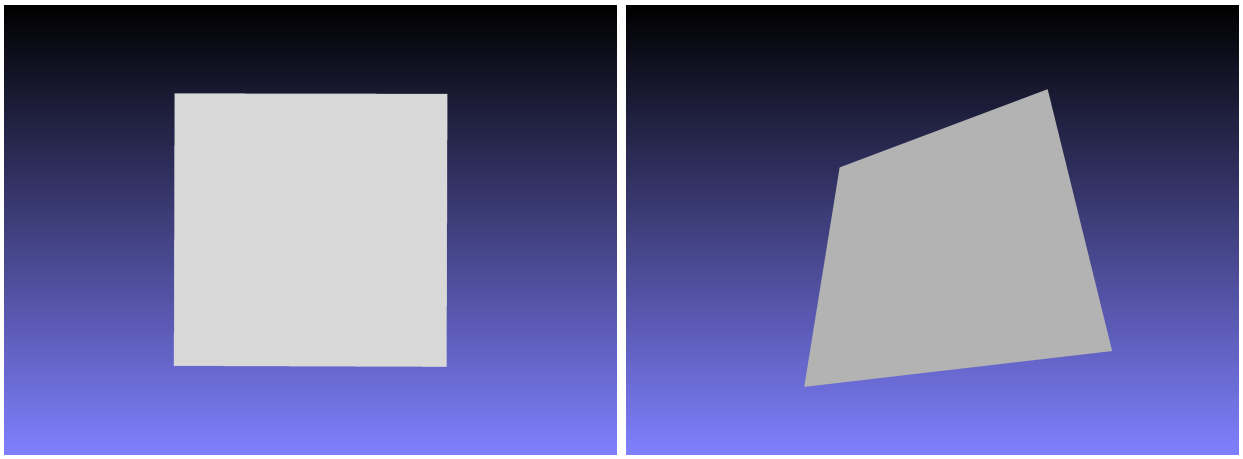
The discretized Laplace-Beltrami operator using cotangent discretization may enhance the detail of curvature estimation in complex models (like the Lilium); it could also introduce unexpected artefacts in simpler geometries (like the plane), which suggests that careful interpretation and potential further smoothing may be required for consistent curvature approximation.



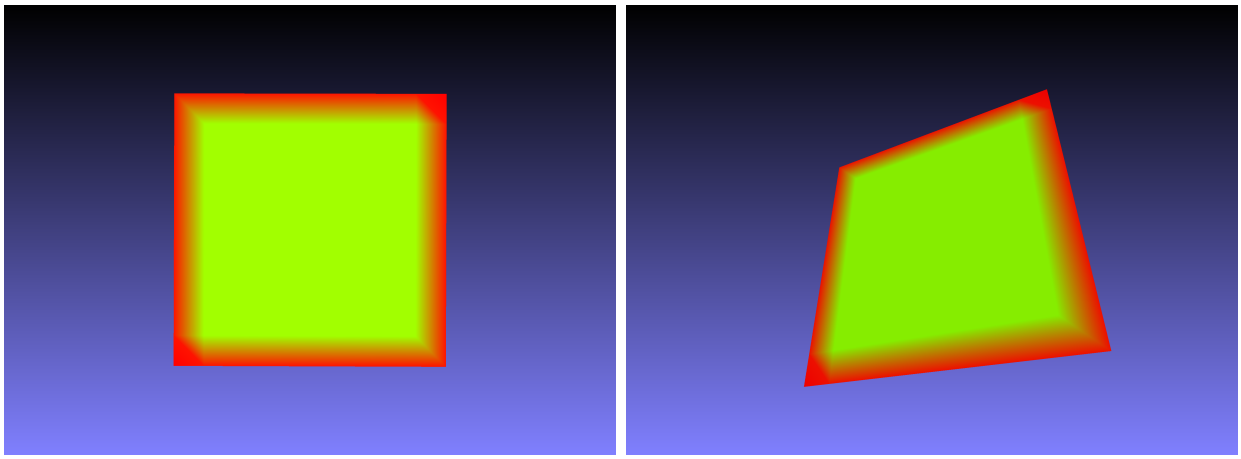
(a) The first viewpoint of the original mesh (b) The second viewpoint of the original mesh

(c) The first viewpoint of the mean curvature (d) The second viewpoint of the mean curvature

Figure 2: Visualization results of liliums.



(a) The first viewpoint of the original mesh (b) The second viewpoint of the original mesh



(c) The first viewpoint of the mean curvature (d) The second viewpoint of the mean curvature

Figure 3: Visualization results of Plane.

Question 4 Modal analysis:

As k increases, we expect the reconstructed mesh to more closely approximate the original mesh geometry. This is due to the higher number of eigenvectors capturing more details of the mesh surface. The comparison in Fig .4of reconstructions for $k = 5$, $k = 15$, and a larger k should show progressive improvements in detail and accuracy.

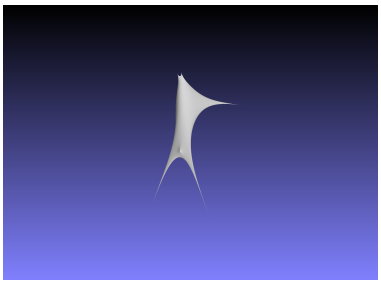
Question 5 Explicit Laplacian Mesh Smoothing:

Based on Lecture 6, we first implemented the uniform Laplace-Beltrami operator:

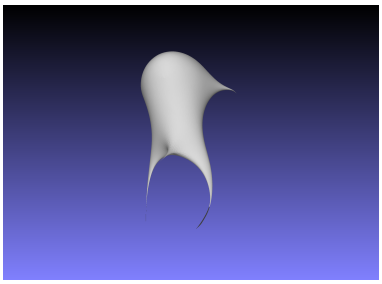
$$L = \frac{1}{|\mathcal{N}_1(v_i)|} \sum_{v_j \in \mathcal{N}_1(v_i)} (f(v_j) - f(v_i)).$$

Then, we utilized this operator to iteratively smooth each vertex X^n:

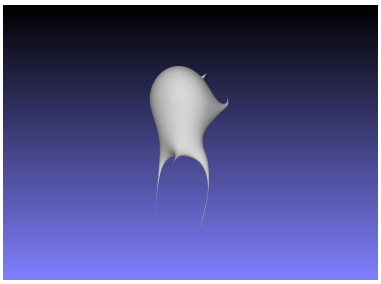
$$X^{n+1} = X^n + \lambda L X^n.$$



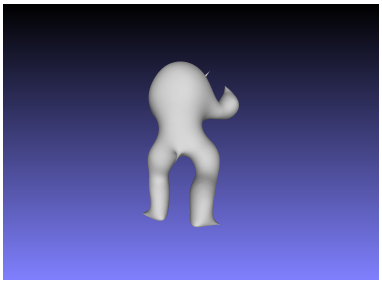
(a) $k = 5$ eigenvectors



(b) $k = 10$ eigenvectors



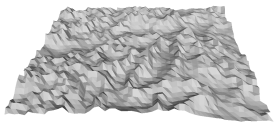
(c) $k = 15$ eigenvectors



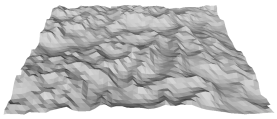
(d) $k = 100$ eigenvectors

Figure 4: Comparison of mesh reconstructions with varying numbers of eigenvectors.

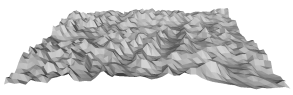
We conducted experiments on *plane_ns.obj*. We set the number of iterations uniformly to 50 and lambda to $\{0.01, 0.05, 1, 2\}$.



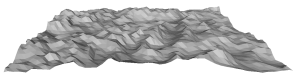
(a) The first viewpoint of the original mesh



(b) The first viewpoint of the $\lambda = 0.01$

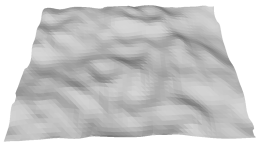


(c) The second viewpoint of the original mesh



(d) The second viewpoint of the $\lambda = 0.01$

Figure 5: Visualization results of explicit Laplacian mesh smoothing under different λ.



(a) The first viewpoint of the $\lambda = 0.05$

(b) The first viewpoint of the $\lambda = 1$

(c) The second viewpoint of the $\lambda = 0.05$

(d) The second viewpoint of the $\lambda = 1$

Figure 6: Visualization results of explicit Laplacian mesh smoothing under different λ.

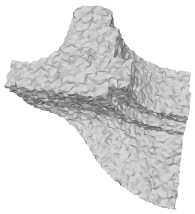
(a) The first viewpoint of the $\lambda = 2$

(b) The second viewpoint of the $\lambda = 2$

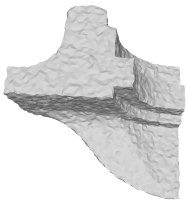
Figure 7: Visualization results of explicit Laplacian mesh smoothing under excessive λ.

As shown in Fig. 5 and 6, when $\lambda = 0.01$, there is a certain level of smoothing effect, and when $\lambda = 0.05$, the smoothing effect is more pronounced while retaining the original mesh features. However, when $\lambda = 1$, excessive smoothing occurs, leading to loss of the original mesh features. Furthermore, with a lambda that is too large, for example, $\lambda = 2$ as shown in Fig. 7, the mesh shape becomes deformed. In summary, the choice of λ should not be too large; a value around 0.05 yields the best smoothing effect.

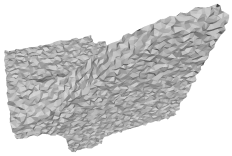
Next, we validate our conclusion on the *fandisk.obj* dataset. As shown in Fig. 8 and Fig. 9, when $\lambda = 0.05$, the smoothing effect and preservation of features are optimal. However, selecting a λ that is too large will result in a change in the shape of the original mesh.



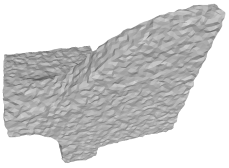
(a) The first viewpoint of the original mesh



(b) The first viewpoint of the $\lambda = 0.01$



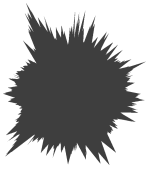
(c) The second viewpoint of the original mesh



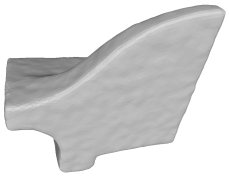
(d) The second viewpoint of the $\lambda = 0.01$

Figure 8: Visualization results of explicit Laplacian mesh smoothing under different λ.

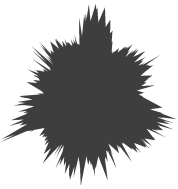
(a) The first viewpoint of the $\lambda = 0.05$



(b) The first viewpoint of the $\lambda = 2$



(c) The second viewpoint of the $\lambda = 0.05$



(d) The second viewpoint of the $\lambda = 2$

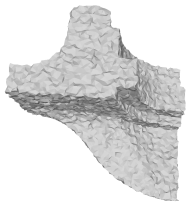
Figure 9: Visualization results of explicit Laplacian mesh smoothing under different λ.

Question 6 Implicit Laplacian Mesh smoothing:

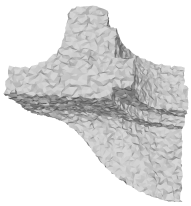
Following the work [4], we implemented implicit Laplacian mesh smoothing using Eq(9):

$$(I - \lambda L)X_8^{(n+1)} = X^n.$$

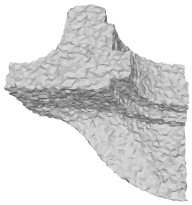
We conducted evaluations on implicit Laplacian mesh smoothing from two perspectives, namely λ and the number of iterations. Firstly, we set the number of iterations to 50, and λ to 0.01, 0.05, 0.1, 2, 10, 100.



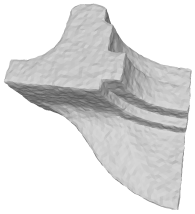
(a) The first viewpoint of the $\lambda = 0.01$



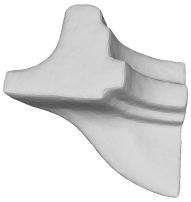
(b) The first viewpoint of the $\lambda = 0.05$



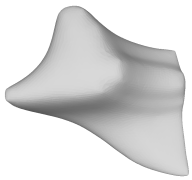
(c) The second viewpoint of the $\lambda = 0.1$



(d) The second viewpoint of the $\lambda = 2$



(e) The second viewpoint of the $\lambda = 10$



(f) The second viewpoint of the $\lambda = 100$

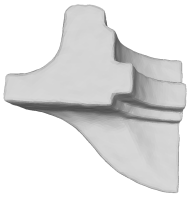
Figure 10: Visualization results of implicit Laplacian mesh smoothing under different λ.

As shown in Fig. 10, the implicit Laplacian mesh smoothing method exhibits better stability. When facing larger values of λ, the algorithm only induces deformation at the edges rather than global deformation. The optimal λ is approximately around 5.

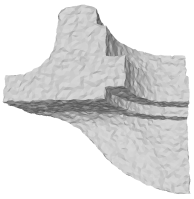
Next, we explore the convergence speed of the algorithm. We set λ to 5 and compare it with the explicit Laplacian mesh smoothing method. We iterate 10 times and 50 times respectively. The results are shown in Fig. 11. We observe that when iterated 10 times, the explicit Laplacian mesh smoothing method has not yet converged, while the implicit Laplacian mesh smoothing method has already exhibited a smooth mesh. In summary, the implicit Laplacian mesh smoothing method demonstrates both better stability and faster convergence speed.

(a) Implicit Laplacian with 50 iterations

(b) Implicit Laplacian with 10 iterations



(c) Explicit Laplacian with 50 iterations



(d) Explicit Laplacian with 10 iterations

Figure 11: Visualization results of implicit and explicit Laplacian under different iterations.

Question 7 Evaluate the Performance

We added Gaussian noise $\mathcal{N}(0, \sigma)$ to the vertices of the original mesh, where σ belongs to $\{0.05, 0.1, 0.5\}$, and compared explicit Laplacian mesh smoothing method and implicit Laplacian mesh smoothing method respectively. From Fig. 12, it can be seen that when sigma is low, such as $\sigma = 0.05$ and $\sigma = 0.1$, the noisy mesh still retains the shape of the original mesh, indicating that Laplacian mesh smoothing maintains its desired effect. When $\sigma = 0.5$, the noisy mesh has already lost the shape of the original mesh, so in this case, the Laplacian mesh smoothing method will not work.

References

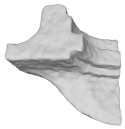
[1] "Numpy documentation." `https://numpy.org/doc/`

[2] "Matplotlib documentation." `https://matplotlib.org/`

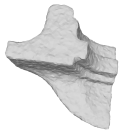
[3] "Trimesh documentation." `https://trimsh.org/`

[4] Desbrun, M., Meyer, M., Schröder, P. & Barr, A. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings Of The 26th Annual Conference On Computer Graphics And Interactive Techniques*. pp. 317-324 (1999)

(a) The noisy with $\sigma = 0.05$

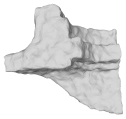


(b) The explicit method

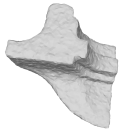


(c) The implicit method

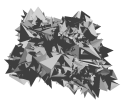
(d) The noisy with $\sigma = 0.1$



(e) The explicit method



(f) The implicit method



(g) The noisy with $\sigma = 0.5$

(h) The explicit method

(i) The implicit method

Figure 12: Visualization results of implicit Laplacian mesh smoothing under different σ.