1 Uniform Laplace

Figure 1: Color Map

In this report viridis style colormap is adopted to show degree of curvature as figure 1.

1.1 Mean Curvature

$$\Delta_{\text{uni}} f(v_i) := \frac{1}{|\mathcal{N}_1(v_i)|} \sum_{v_j \in \mathcal{N}_1(v_i)} (f(v_j) - f(v_i)) \tag{1}$$

Mean curvature can be computed as (1) which get differences of neighbours and average it by number of neighbours. In implementation, method "vertex_neighbors" from trimesh library is used to get neighbour vertices. This method has benefit of its simplicity and efficiency but it has bad approximation for irregular triangulations. We can find it works well on simple mesh as figure 2.

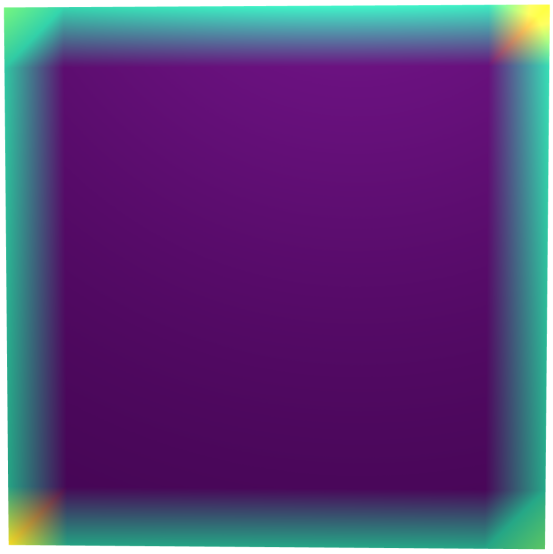


Figure 2: Mean Curvature plane.obj

1.2 Gaussian Curvature

$$K = \frac{(2\pi - \sum_j \theta_j)}{A}$$

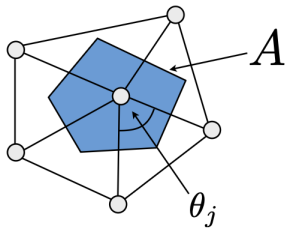


Figure 3: Gaussian curvature equation

Gaussian curvature K can be computed as shown in Figure 6 by using the area of vertices as depicted in the image and the sum of the angles. In the implementation, the method `face_angles` and `area_faces` from the trimesh library [1] is used to obtain the angle θ area A. The result of apply Gaussian curvature for plane.obj can be found on figure 4.

Figure 4: Gaussian Curvature for plane.obj

1.3 Comparison and Discussion

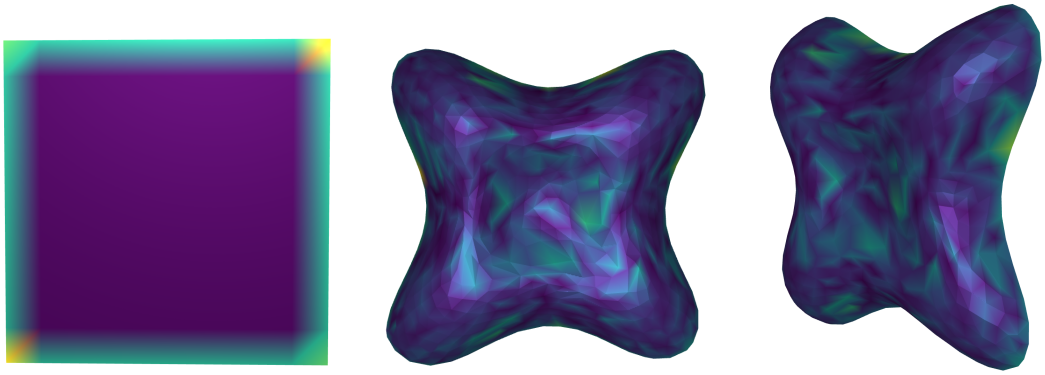
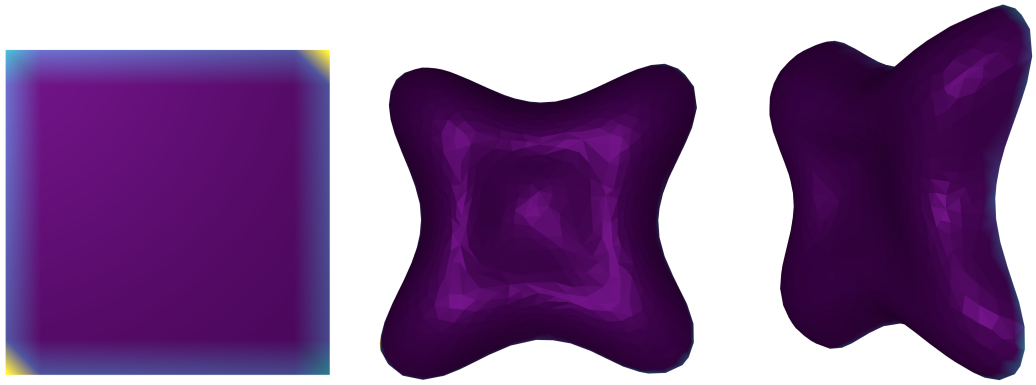
	Plane	Lilium Front	Lilium Side
Mean			
Gaussian			

Table 1: Comparison of Gaussian and Mean Curvatures from Various Views

Mean curvature shows rapid changes in regions where the surface have significant bending, marking sharp changes that accurately highlight areas of high curvature intensity. This characteristic makes mean curvature particularly sensitive and effective in detailing the local variations in the geometry of the surface, offering a granular view of curvature at individual vertices.

On the contrast, Gaussian curvature presents a smoother transition, especially around the boundaries. This change in boundary is because of discontinuous of mesh. This smoother change means that Gaussian curvature is good for capturing the overall curvature behavior. Therefore, both of them have different strength for computing curvature, but it is uniformly calculated, so may not work well for complex meshes.

2 First and Second Fundamental forms

To compute the normal curvature at the point $(a, 0, 0)$, we need to use the first and second fundamental forms. u and v at $(a, 0, 0)$ can be represented as 0 and π. Based on the parametric equation of the ellipsoid, we can calculate matrix I and II the which use partial derivatives and

second derivative respectively as follow:

$$\mathrm{I} = \begin{pmatrix} \mathbf{x}_u^T \mathbf{x}_u & \mathbf{x}_u^T \mathbf{x}_v \\ \mathbf{x}_u^T \mathbf{x}_v & \mathbf{x}_v^T \mathbf{x}_v \end{pmatrix}$$

$$\mathrm{II} = \begin{pmatrix} \mathbf{x}_{uu}^T \mathbf{n} & \mathbf{x}_{uv}^T \mathbf{n} \\ \mathbf{x}_{uv}^T \mathbf{n} & \mathbf{x}_{vv}^T \mathbf{n} \end{pmatrix}$$

Based on First and Second fundamental form I and II, we can compute normal curvature as following formula where $\bar{\mathbf{t}}$ is unit vector.

$$\kappa_n(\bar{\mathbf{t}}) = \frac{\bar{\mathbf{t}}^T \mathrm{II} \bar{\mathbf{t}}}{\bar{\mathbf{t}}^T \mathrm{I} \bar{\mathbf{t}}}$$

Based on this calculation, we can obtain Figure 5 which shows the highest curvature at $0, \pi, 2\pi$ and the lowest curvature at $\frac{\pi}{2}, \frac{3\pi}{2}$.

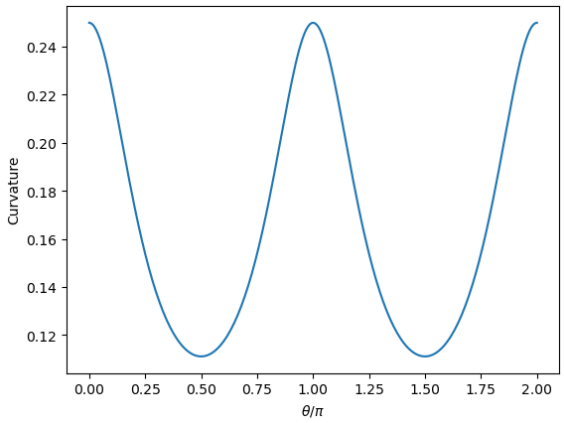


Figure 5: Normal curvature κ_n at $(a, 0, 0)$

3 Non-uniform (Discrete Laplace-Beltrami)

$$\Delta_S f(v_i) := \frac{1}{2A(v_i)} \sum_{v_j \in N_1(v_i)} (\cot \alpha_{ij} + \cot \beta_{ij})(f(v_j) - f(v_i))$$

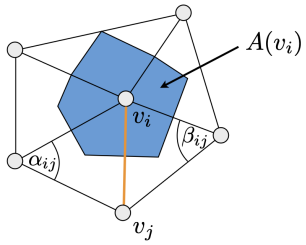


Figure 6: Cotangent Discretization

The main difference of this method from those discussed previously lies in its weighting scheme. Specifically, the weight for each vertex is calculated using the cotangent of the angles formed by the edges of the adjacent triangles, which can be derived from the dot product of the vectors as follows:

$$\cos(\alpha) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

This cotangent weighting, expressed as $\cot\alpha + \cot\beta$, adapts to the local geometry, leading to a more accurate approximation of the continuous operator. For stability, any negative weights are treated as zero as mentioned by [2]. This adjustment ensures that the discretization remains well-behaved in practice, avoiding potential numerical instabilities.

As shown in Figure 7, the plane demonstrates a less rapid change in curvature compared to mean curvature, yet exhibits a more pronounced change than Gaussian curvature. Specifically, for the Lilium object, this approach yields high curvature values in areas that are highly curved, distinguishing itself from mean curvature, which tends to indicate high curvature across most parts, and Gaussian curvature, which typically shows low curvature except at discontinuities. This result shows that it is more efficient than the other two methods for computing complex objects since it assigns weights to each vertex based on the local geometric configuration. This weighting allows for a adaptation to the complex object's surface, therefore providing a more precise measure of curvature.

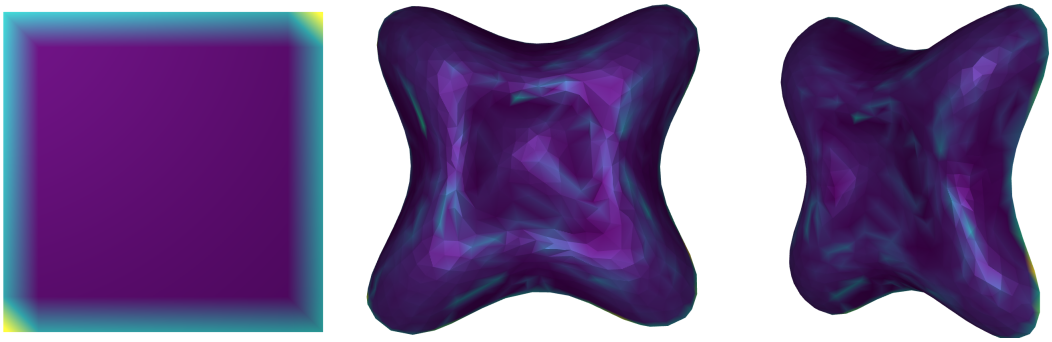


Figure 7: Cotangent discretization mesh smoothing with plane and lilium

4 Modal Analysis

The discrete Laplace-Beltrami operator is defined by the matrix C with entries C_{ij} computed as:

$$C_{ij} = \begin{cases} \frac{\cot\alpha_{ij}+\cot\beta_{ij}}{A_i+A_j} > 0, & \text{if } i \neq j \text{ and } j \in \mathcal{N}_1(v_i) \\ -\sum_{v_j \in \mathcal{N}_1(v_i)} \frac{\cot\alpha_{ij}+\cot\beta_{ij}}{A_i+A_j}, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where A_i and A_j represent the areas associated with vertices i and j, respectively.

In modal analysis, the weights that obtain negative values are filtered out as mentioned [2]. Different from Task 3, the weights are normalized by the areas of vertices i and j to improve mesh reconstruction, as suggested by [3].

$$\mathbf{x} := [x_1, \ldots, x_n] \qquad \mathbf{y} := [y_1, \ldots, y_n] \qquad \mathbf{z} := [z_1, \ldots, z_n]$$

$$\mathbf{x} \leftarrow \sum_{i=1}^{k} \left(\mathbf{x}^T \mathbf{e}_i\right) \mathbf{e}_i \qquad \mathbf{y} \leftarrow \sum_{i=1}^{k} \left(\mathbf{y}^T \mathbf{e}_i\right) \mathbf{e}_i \qquad \mathbf{z} \leftarrow \sum_{i=1}^{k} \left(\mathbf{z}^T \mathbf{e}_i\right) \mathbf{e}_i$$

Figure 8: Reconstruction Equation

After constructing the discrete Laplace-Beltrami operator, the mesh can be reconstructed using the K smallest eigenvectors of the Laplace-Beltrami matrix L as shown in Equation 8.

As observed from Figure 9, with an increasing number of K, the result of the reconstruction becomes more defined and closely resembles the original mesh shape. Especially end part of mesh such as hand, feet and tail shows biggest change as number of eigenvectors increases. This improvement is due to the inclusion of finer geometric details captured by the higher-order eigenvectors.

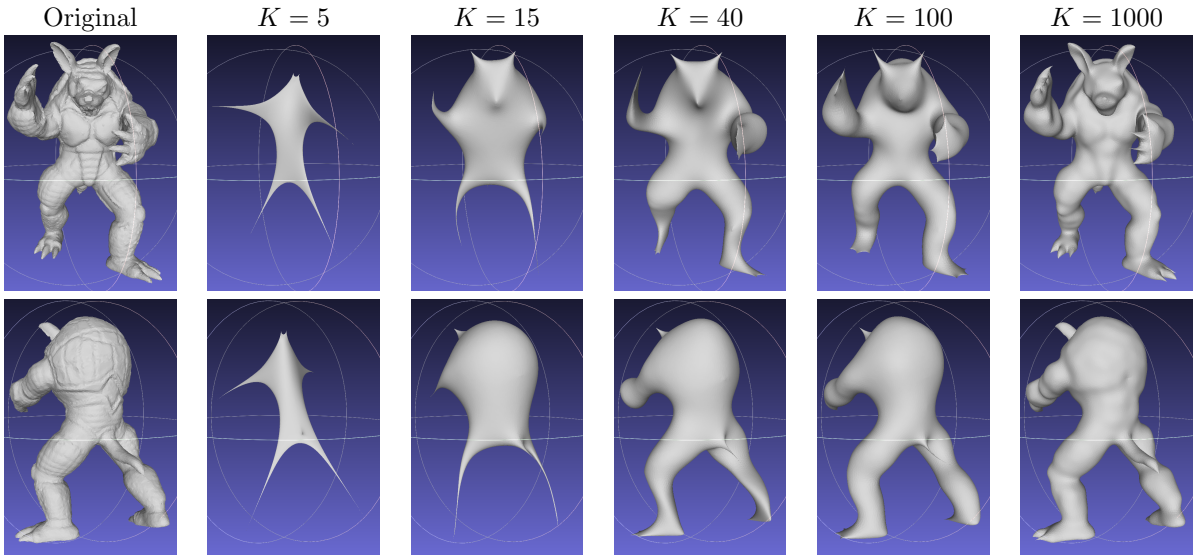


Figure 9: Mesh reconstructions with different number of K

5 Explicit Laplacian Mesh Smoothing

$$P(t+1) = (I + \lambda\Delta)P(t) \tag{3}$$

For implementing explicit Laplacian Mesh smoothing, we can simply use equation (3). Value of λ is important for good representation. As we can see figure 10 too low value of λ cannot give enough smoothing effect and too high value of λ smooth also important feature such as edges and vertices. For plane, 1 is good value to choose since it does not have many feature, but for fandisk, 0.5 is good value to choose since it has more features and they have to be preserved.

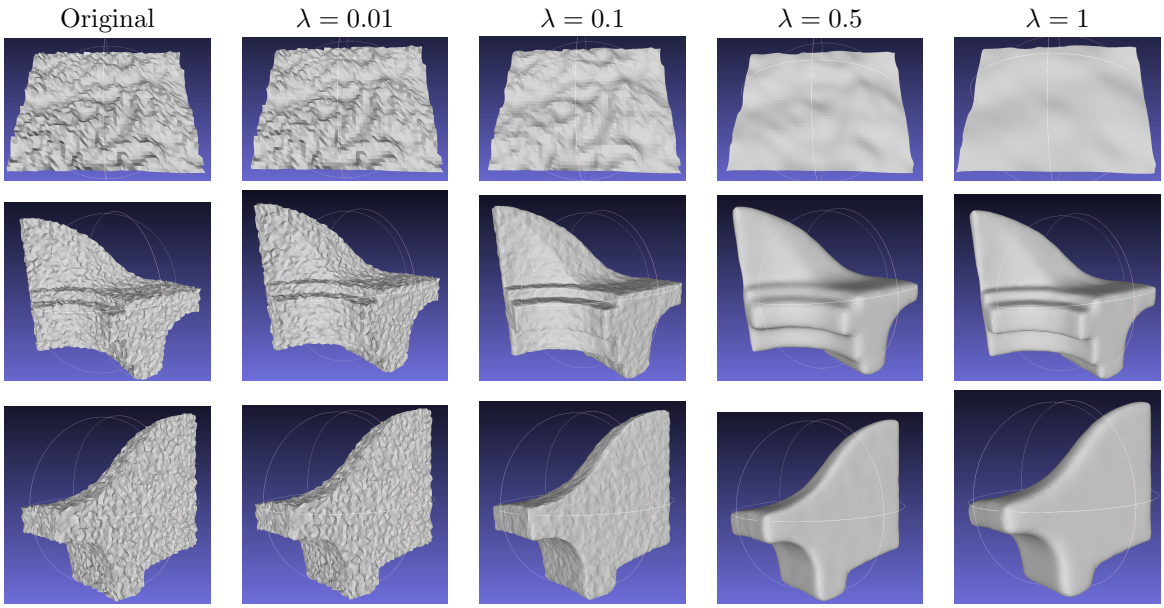


Figure 10: Explicit smoothing with different λ with 10 iteration

However, using too high value of λ gives distortion as figure 11. It is because if λ is large, the term $\lambda \Delta P(t)$ can dominate the original position $P(t)$ in the equation. Vertices may move too far, potentially inverting the mesh if they overshoot their "ideal" new position.

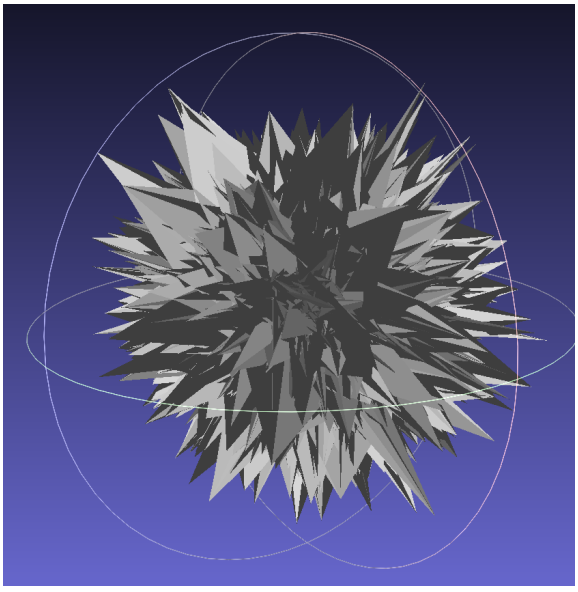


Figure 11: Mean Curvature plane.obj

6 Implicit Laplacian Mesh Smoothing

$$(I - \lambda\Delta)P(t+1) = P(t) \tag{4}$$

In contrast to explicit Laplacian mesh smoothing, which directly computes the new positions of vertices based on the current positions and a laplacian matrix, implicit Laplacian mesh smoothing [4] employs an indirect approach to update vertex positions. This method is described by the equation 4. Similar as task 5, for plane, 1 is good value to choose since it does not have many feature, but for fandisk, 0.5 is good value to choose since it has more features and they have to be preserved as we can find from figure 12.

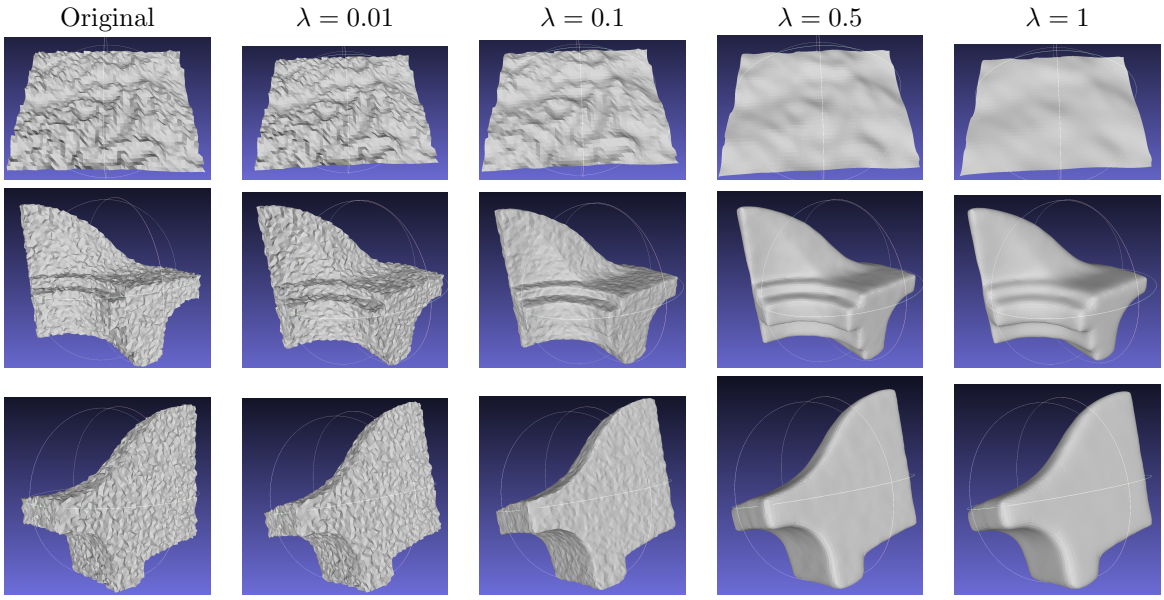


Figure 12: Implicit smoothing with different λ with 10 iteration

Furthermore, as we can find from figure 13, as iteration increases the amount of smoothing also increases. Therefore, from this experiment we can find increasing number of iteration can give similar effect as increasing number of λ.

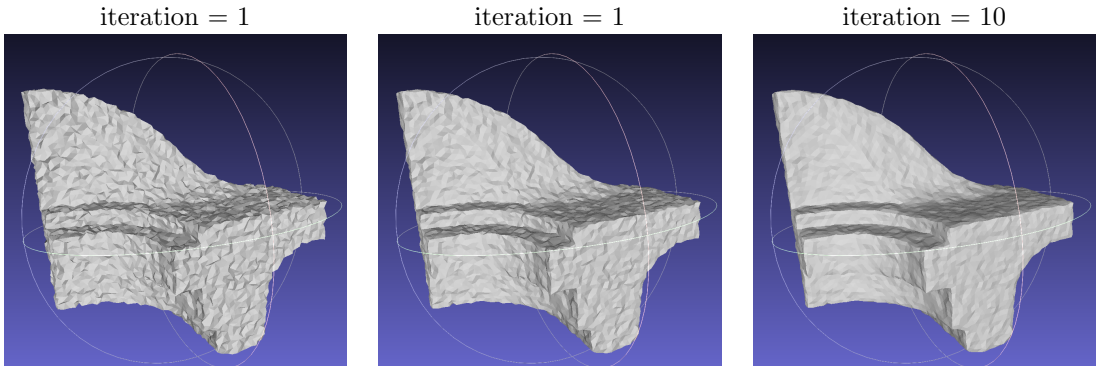


Figure 13: Different iteration with $\lambda = 0.1$

The key difference between the explicit and implicit mesh smoothing methods lies in their stability, particularly noticeable when employing larger λ values. It is because implicit mesh smoothing achieves unconditional stability and quickly reaches the PDE equilibrium by using a time-shifted evaluation based on the mesh's future state as mentioned by [4]. This stability difference is illustrated in figure 14, where, unlike its explicit counterpart, the implicit method demonstrates stability against distortion even at high λ levels. This stability of the implicit approach permits the use of larger time steps without encountering the mesh distortion commonly associated with high λ values in explicit smoothing. Essentially, the implicit method serves as a robust filter, capable of accommodating a wider range of λ values. This adaptability significantly minimizes the risk of vertices overshooting their intended positions, thereby preventing mesh inversion and preserving the integrity of the geometric structure

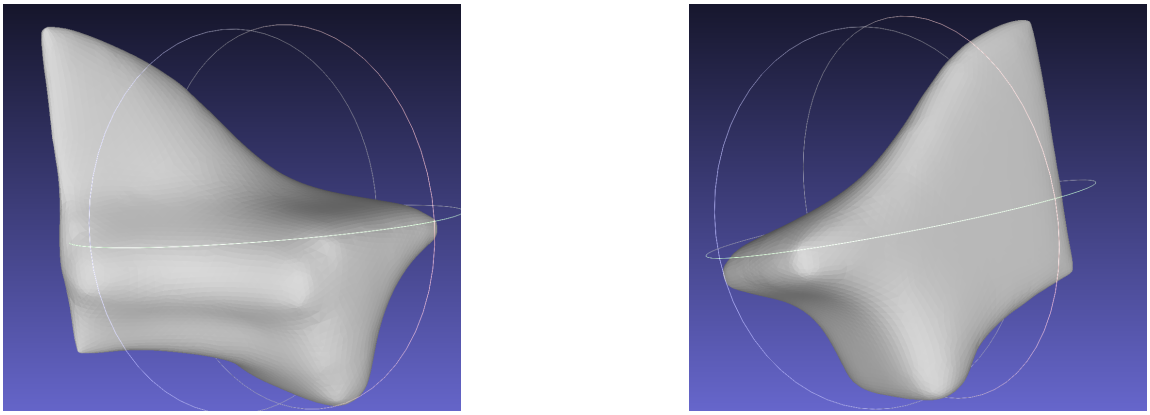


Figure 14: Implicit smoothing with large $\lambda = 5$

7 Laplacian mesh denoising

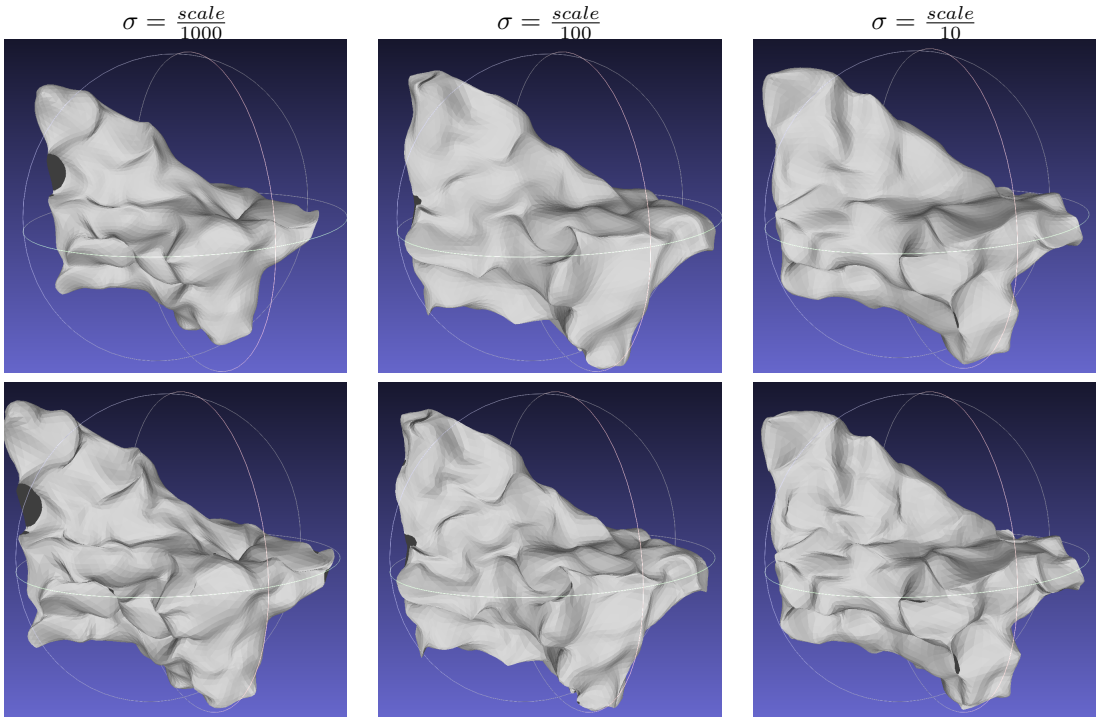


Figure 15: Mesh denoising with different amount of noise. (Up: Explicit, Down: Implicit)

As illustrated in Figure 15 which shows smoothing effect with different sigma with scale of mesh from [1], the addition of noise to the noisy object needs a reevaluation of the parameter λ. The optimal value for λ, which is 1 for the standard fandisk_ns, fails to give satisfactory results under noisier conditions. This observation suggests that a larger value of λ is required to improve denoising outcomes. However, explicit smoothing methods showed that a λ exceeding 1 leads to distortions. Therefore, to effectively denoise objects as the noise level increases, one must resort to implicit smoothing techniques with a λ value greater than 1, as depicted in Figure 16.

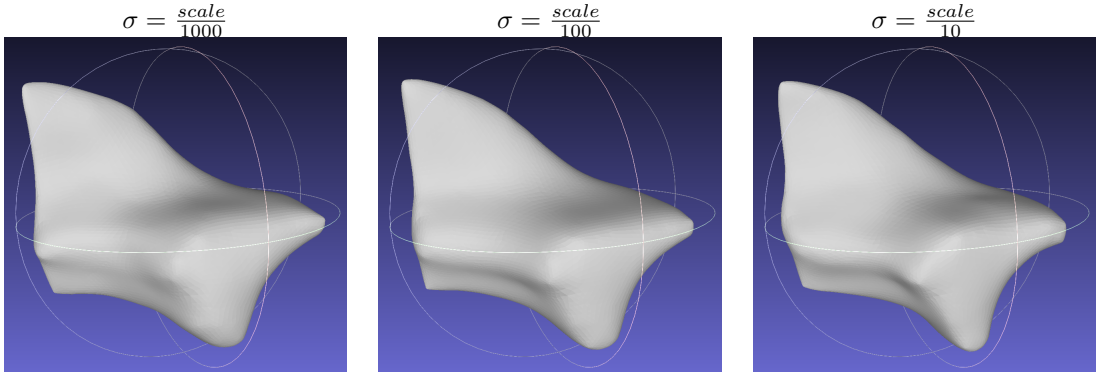


Figure 16: Mesh denoising with Implicit smoothing with $\lambda = 5$.

Moreover, as we employed uniform Laplacian mesh smoothing, we encounter discontinuities within the mesh. Also, considerably high value of λ is needed to achieve satisfactory results. In contrast, as illustrated in Figure 17, utilizing the discrete Laplace-Beltrami operator yields significantly enhanced outcomes at lower λ values. This improvement is attributed to the weighted nature of the discrete Laplace-Beltrami operator, which more effectively preserves the mesh's geometric details while smoothing.

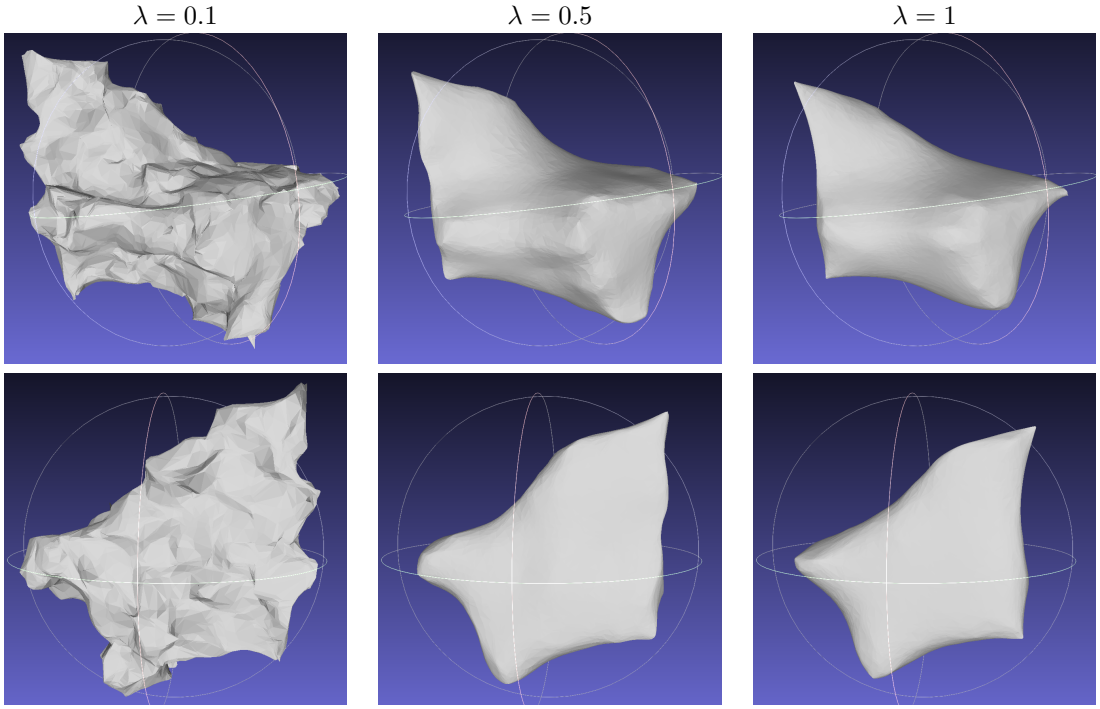


Figure 17: Implicit discrete mesh smoothing with different *lambda* with $\sigma = \frac{scale}{10}$.

References

[1] Dawson-Haggerty et al., "trimesh." [Online]. Available: https://trimesh.org/

[2] B. Levy, "Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry," in *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, 2006, pp. 13–13.

[3] B. Vallet and B. Levy, "Spectral geometry processing with manifold harmonics," *Computer Graphics Forum*, vol. 27, pp. 251 – 260, 04 2008.

[4] P. Schrder, M. Desbrun, M. Meyer, P. Schröder, and A. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," *SIGGRAPH*, vol. 99, 09 2000.