COMP0119 Acquisition and Processing of 3D Geometry Coursework 2

L.C.

2025/03/20

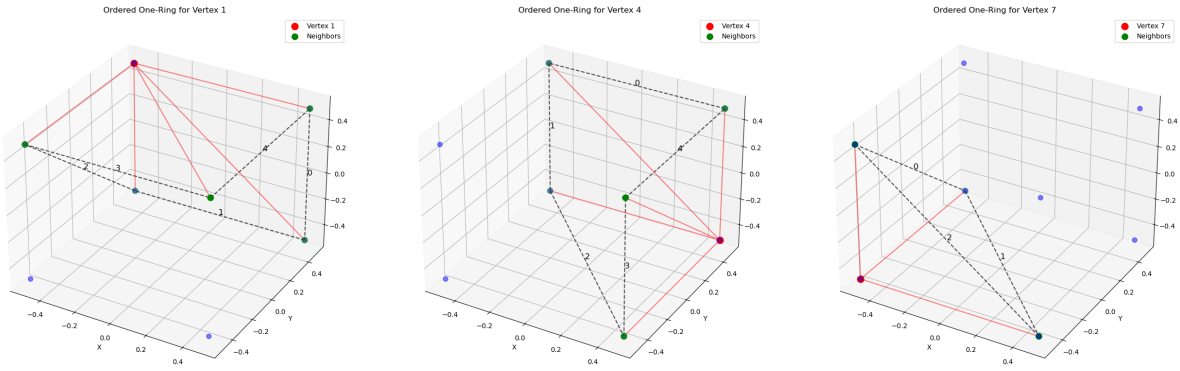
1 Discrete Curvature and Spectral Meshes

This section explores various techniques for computing and analyzing discrete curvature on meshes, as well as spectral mesh processing. We begin with developing tools for structured traversal of mesh elements, then progress to curvature estimation, fundamental form analysis, and ultimately spectral reconstruction of meshes.

1.1 Ordered One-ring Neighborhood

The one-ring neighborhood of a vertex is a fundamental structural element in mesh processing, consisting of all vertices directly connected to a central vertex by an edge. While mesh data structures typically provide neighborhood information, this information lacks consistent ordering, which is crucial for many geometry processing algorithms.

We implemented an algorithm to organize one-ring neighbors in a consistent clockwise or counter-clockwise order. The key insight is to trace through the triangles containing the central vertex, using the mesh connectivity to establish proper ordering:



(a) Vertex 1 (b) Vertex 4 (c) Vertex 7

Figure 1: Ordered one-ring visualization for vertices on a cube mesh, showing the consistent traversal order of neighboring vertices.

The algorithm starts with an arbitrary neighbor, then methodically traverses the triangles containing the central vertex, maintaining the ordering as it progresses. This approach ensures a consistent orientation that respects the local topology of the mesh.

To validate the implementation, we tested the algorithm on a cube mesh, where the expected ordering can be easily verified. Figure 1 demonstrates the algorithm correctly ordering the neighbors of a corner vertex in a clockwise manner.

1.2 Uniform Laplacian and Curvature Estimation

We implemented two fundamental differential operators for mesh geometry: mean curvature and Gaussian curvature.

1.2.1 Mean Curvature via Uniform Laplacian

Mean curvature (H) at a vertex can be computed using the uniform Laplacian operator. For a vertex v_i, the uniform Laplacian is defined as:

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

where $\mathcal{N}_1(v_i)$ represents the one-ring neighborhood of vertex v_i.

The relationship between the Laplace-Beltrami operator and mean curvature is given by:

$$\Delta \mathbf{x} = -2H\mathbf{n} \tag{2}$$

where $\mathbf{x}$ is the position vector, H is the mean curvature, and $\mathbf{n}$ is the unit normal.

Therefore, the mean curvature at vertex v_i is computed as:

$$H(v_i) = -\frac{1}{2}(\Delta \mathbf{x}_i \cdot \mathbf{n}_i) \tag{3}$$

1.2.2 Gaussian Curvature via Angle Deficit

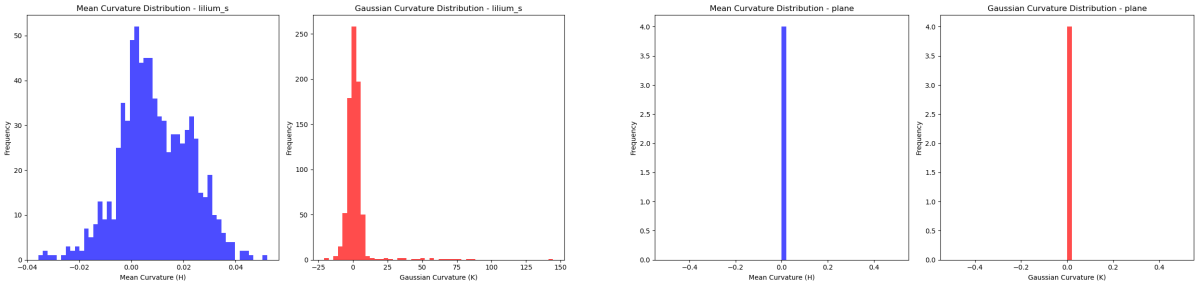
For Gaussian curvature (K), we use the angle deficit formula:

$$K(v_i) = \frac{2\pi - \sum_j \theta_j}{A_i} \tag{4}$$

where θ_j are the angles of the triangles incident to vertex v_i, and A_i is the area associated with the vertex. We use the barycentric cell area, assigning one-third of each adjacent triangle's area to the vertex:

$$A_i = \frac{1}{3} \sum_{f \in F_i} A_f \tag{5}$$

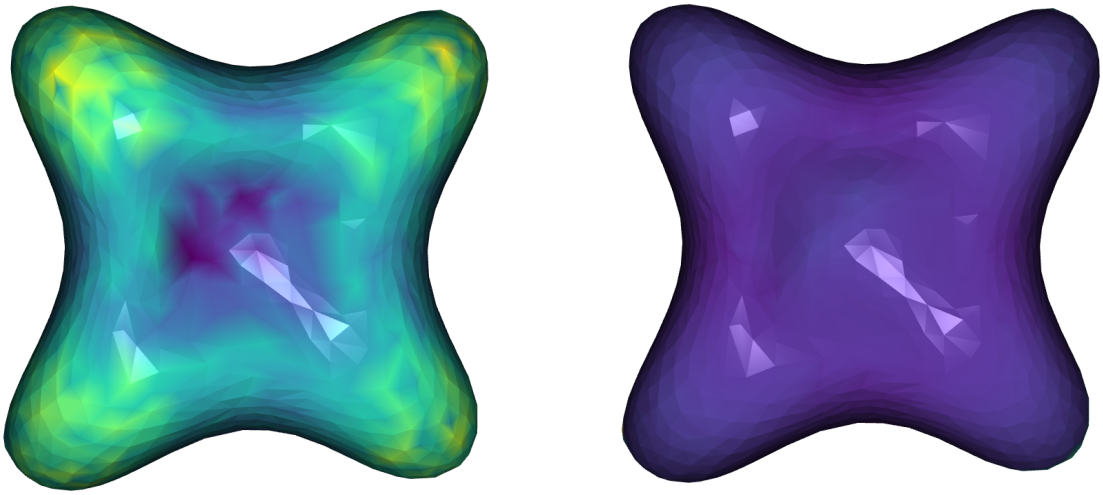
where F_i is the set of faces incident to vertex v_i, and A_f is the area of face f.



(a) Curvature histogram for Lilium model

(b) Curvature histogram for Plane model

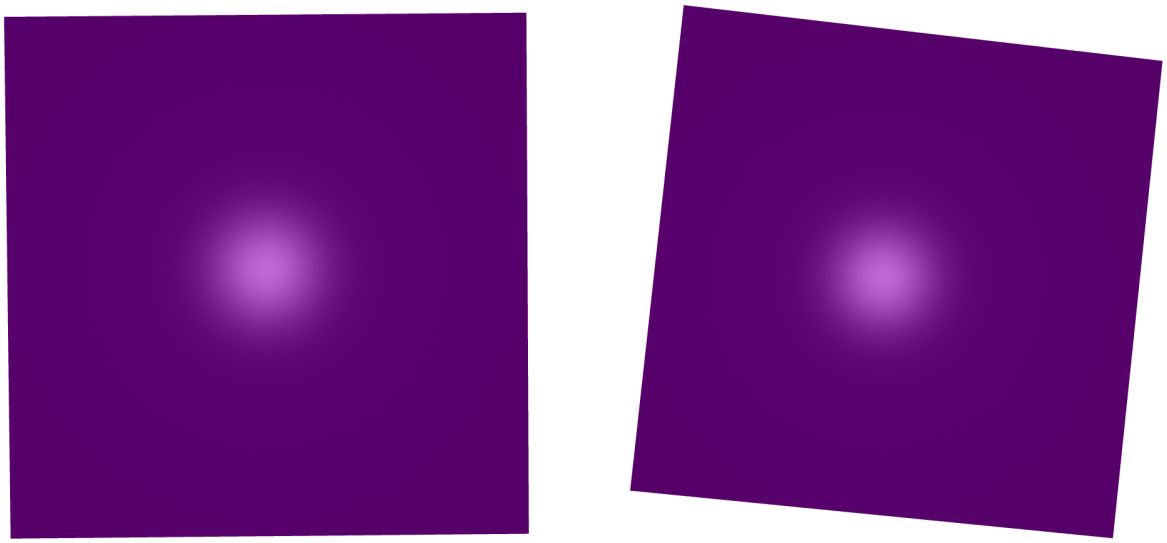
Figure 2: Histograms showing the distribution of mean and Gaussian curvature values for different mesh models.



(a) Mean curvature for Lilium model

(b) Gaussian curvature for Lilium model

Figure 3: Visualization of mean and Gaussian curvatures on Lilium model using the uniform Laplacian.



(a) Mean curvature for Plane model

(b) Gaussian curvature for Plane model

Figure 4: Visualization of mean and Gaussian curvatures on Plane model using the uniform Laplacian.

1.2.3 Analysis of Curvature Estimation

The implemented curvature estimation was tested on two meshes: a complex flower-like model (lilium) and a simple plane. The curvature distributions showed expected patterns:

- On the lilium model, mean curvature values ranged from -0.036 to 0.052, with negative values indicating concave regions and positive values indicating convex regions.
- Gaussian curvature showed a much wider range from -20.75 to 144.55, with negative values identifying saddle points and high positive values identifying regions of significant local curvature.
- For the plane model, both curvature measures were correctly calculated as approximately zero, confirming the accuracy of our implementation.

The histograms of curvature distributions revealed that for the lilium model, mean curvature values were concentrated around zero with a slight positive bias, while Gaussian curvature was predominantly near zero with a few regions having high positive values.

The uniform Laplacian provides a reasonable approximation of curvature for simple shapes but has limitations:

- It does not consider the geometric properties of the mesh, treating all neighbors equally regardless of the local geometry
- It can be biased in regions where triangle sizes vary significantly
- It may smooth out sharp features due to the averaging nature of the operator

Despite these limitations, the uniform Laplacian-based approach correctly identified key curvature characteristics of both test meshes, demonstrating its utility as a basic curvature estimation method.

1.3 First and Second Fundamental Forms

The differential geometry of surfaces can be characterized by the first and second fundamental forms. For this section, we analyzed the helicoid surface, parameterized as:

$$p(u, v) = (u \cos v, u \sin v, v) \tag{6}$$

where $u \in [0, 1]$ and $v \in [0, 2\pi]$.

1.3.1 First Fundamental Form

The first fundamental form describes how the parameterization distorts distances. It requires computing the partial derivatives:

$$p_u = \frac{\partial p}{\partial u} = (\cos v, \sin v, 0) \tag{7}$$

$$p_v = \frac{\partial p}{\partial v} = (-u \sin v, u \cos v, 1) \tag{8}$$

The coefficients of the first fundamental form are:

$$E = p_u \cdot p_u = \cos^2 v + \sin^2 v = 1 \tag{9}$$

$$F = p_u \cdot p_v = -\cos v \cdot u \sin v + \sin v \cdot u \cos v = 0 \tag{10}$$

$$G = p_v \cdot p_v = u^2 \sin^2 v + u^2 \cos^2 v + 1 = u^2 + 1 \tag{11}$$

Therefore, at any point $p(u, v)$, the first fundamental form is:

$$I = \begin{pmatrix} E & F \\ F & G \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & u^2 + 1 \end{pmatrix} \tag{12}$$

1.3.2 Second Fundamental Form

The second fundamental form describes how the surface curves in the embedding space. We need the second derivatives and the normal vector:

$$p_{uu} = \frac{\partial^2 p}{\partial u^2} = (0, 0, 0) \tag{13}$$

$$p_{uv} = \frac{\partial^2 p}{\partial u \partial v} = (-\sin v, \cos v, 0) \tag{14}$$

$$p_{vv} = \frac{\partial^2 p}{\partial v^2} = (-u \cos v, -u \sin v, 0) \tag{15}$$

The unit normal is:

$$n = \frac{p_u \times p_v}{|p_u \times p_v|} = \frac{(\sin v, -\cos v, u)}{\sqrt{1+u^2}} \tag{16}$$

The coefficients of the second fundamental form are:

$$e = p_{uu} \cdot n = 0 \tag{17}$$

$$f = p_{uv} \cdot n = \frac{-\sin v \cdot \sin v - \cos v \cdot (-\cos v)}{\sqrt{1+u^2}} = \frac{-1}{\sqrt{1+u^2}} \tag{18}$$

$$g = p_{vv} \cdot n = \frac{-u \cos v \cdot \sin v - u \sin v \cdot (-\cos v)}{\sqrt{1+u^2}} = 0 \tag{19}$$

Therefore, the second fundamental form at any point $p(u, v)$ is:

$$II = \begin{pmatrix} e & f \\ f & g \end{pmatrix} = \begin{pmatrix} 0 & \frac{-1}{\sqrt{1+u^2}} \\ \frac{-1}{\sqrt{1+u^2}} & 0 \end{pmatrix} \tag{20}$$

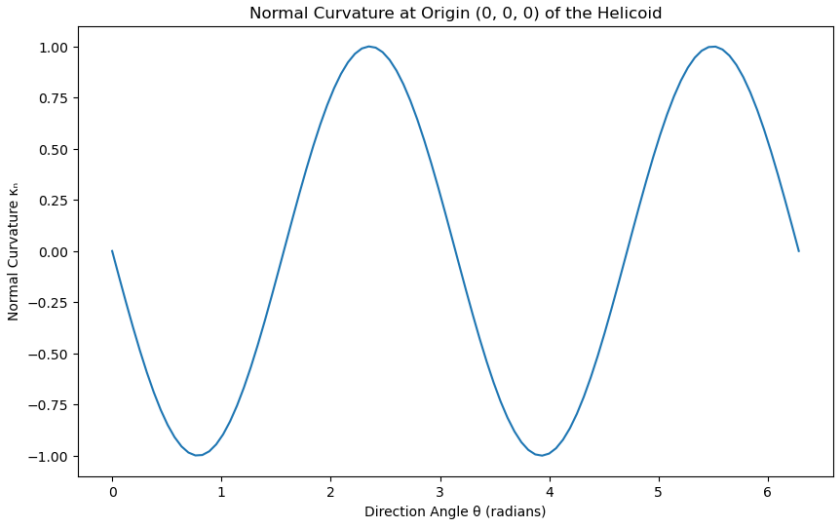


Figure 5: Normal curvature of the helicoid at the origin as a function of direction angle θ, showing the sinusoidal pattern with maximum and minimum values at approximately $\theta = 3\pi/4$ and $\theta = 7\pi/4$.

1.3.3 Normal Curvature Analysis

The normal curvature at a point in direction θ on the tangent plane is:

$$\kappa_n(\theta) = \frac{e\cos^2\theta + 2f\cos\theta\sin\theta + g\sin^2\theta}{E\cos^2\theta + 2F\cos\theta\sin\theta + G\sin^2\theta} \tag{21}$$

At the origin ($u = 0, v = 0$), this simplifies to:

$$\kappa_n(\theta) = \frac{-2\cos\theta\sin\theta}{\cos^2\theta + \sin^2\theta} = -\sin(2\theta) \tag{22}$$

Figure 5 shows the sinusoidal pattern of normal curvature, with maximum and minimum values of ± 1 occurring at approximately $\theta = 3\pi/4$ and $\theta = 5\pi/4$ respectively.

The principal curvatures at the origin are $\kappa_1 = 1$ and $\kappa_2 = -1$, which are attained in directions $\theta \approx 3\pi/4$ and $\theta \approx 5\pi/4$, respectively. These equal but opposite principal curvatures classify the origin as a hyperbolic point (saddle point) with zero mean curvature ($H = (\kappa_1 + \kappa_2)/2 = 0$) but negative Gaussian curvature ($K = \kappa_1 \cdot \kappa_2 = -1$).

1.4 Non-uniform Laplacian (Discrete Laplace-Beltrami)

The uniform Laplacian assigns equal weights to all neighboring vertices, regardless of the local geometry. A more accurate approach is the cotangent Laplacian (Laplace-Beltrami operator), which takes into account the local geometry of the mesh.

1.4.1 Cotangent Weights Formulation

For a function f defined on a manifold, the Laplace-Beltrami operator $\Delta_M f$ can be discretized using the cotangent formula:

$$(\Delta_M f)_i = \frac{1}{2A_i} \sum_{j \in N(i)} (\cot\alpha_{ij} + \cot\beta_{ij})(f_j - f_i) \tag{23}$$

Where:

- A_i is the area associated with vertex i
- $N(i)$ is the set of vertices adjacent to vertex i
- α_{ij} and β_{ij} are the angles opposite to the edge (i, j) in the two triangles sharing this edge
- f_i is the value of function f at vertex i

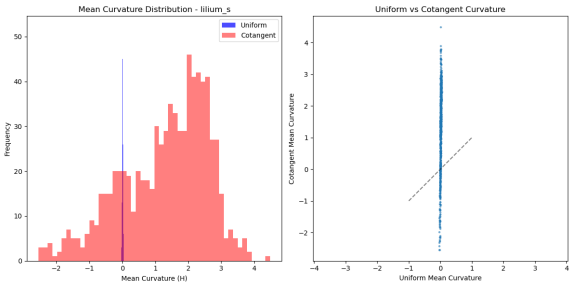
When applied to the coordinate function $\mathbf{x}$, the Laplace-Beltrami operator relates to mean curvature through:

$$\Delta_M \mathbf{x}_i = -2H_i\mathbf{n}_i \tag{24}$$

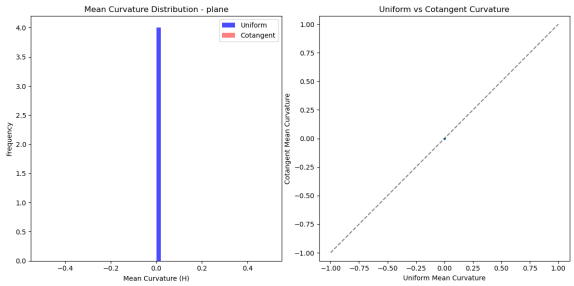
1.4.2 Comparison with Uniform Laplacian

The cotangent Laplacian provides a more accurate estimation of curvature compared to the uniform Laplacian. Key differences observed in our experiments include:

- **Curvature Range:** The cotangent scheme produced a significantly wider range of curvature values compared to the uniform method. For the lilium model, the uniform Laplacian compressed the curvature range to $[-0.036, 0.052]$, while the cotangent method gave $[-2.543, 4.485]$.
- **Feature Preservation:** The cotangent weights better preserved sharp features and captured the true geometric characteristics of the mesh.

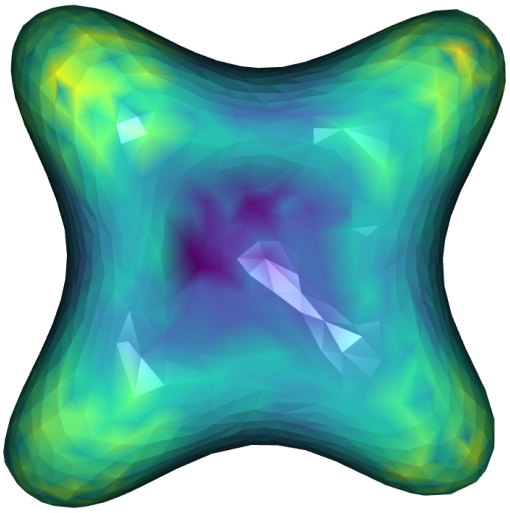


(a) Comparison of curvature estimation methods for Lilium model

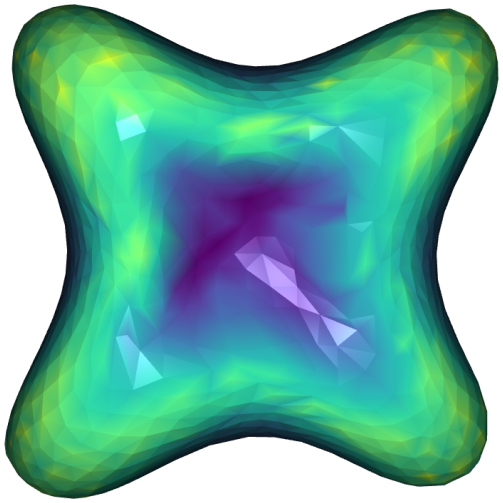


(b) Comparison of curvature estimation methods for Plane model

Figure 6: Comparison of uniform and cotangent Laplacian methods for curvature estimation.



(a) Uniform Laplacian for Lilium model



(b) Cotangent Laplacian for Lilium model

Figure 7: Lilium model Visual comparison of mean curvature estimation using uniform and cotangent Laplacian operators.

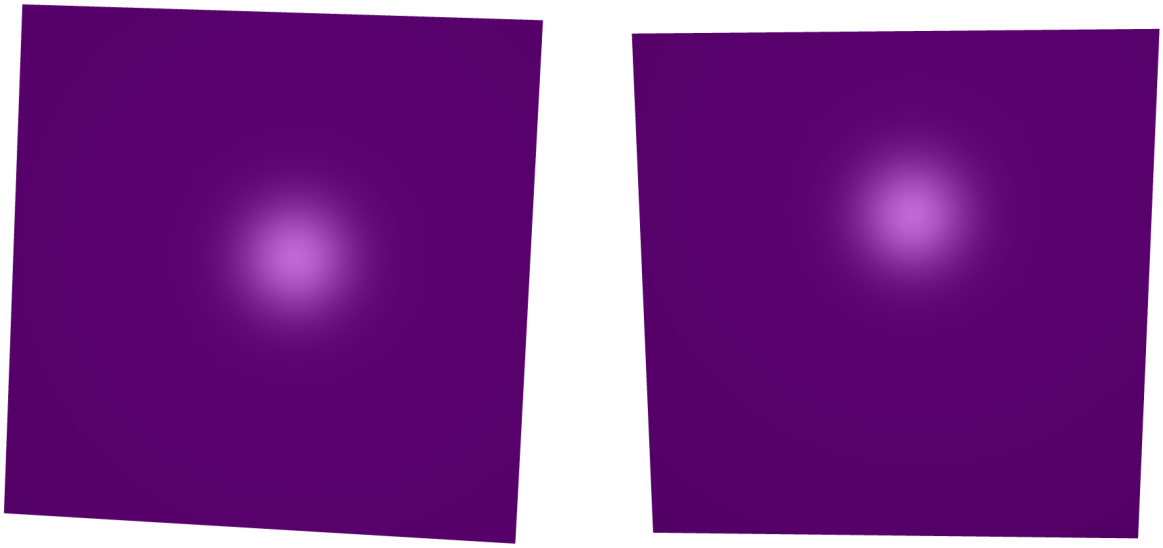
- **Distribution of Values:** Histograms revealed that the uniform Laplacian concentrated most curvature values near zero, while the cotangent Laplacian produced a broader, more detailed distribution.
- **Flat Surface Handling:** Both methods correctly identified zero curvature on the plane mesh, confirming that both are consistent for flat regions.

The significant difference in curvature ranges demonstrates that the cotangent weights are better at capturing the true geometric features of the mesh. This is due to the geometric weighting scheme that accounts for the actual shape of triangles in the mesh, rather than treating all neighbors equally.

As noted by [1], the cotangent Laplacian provides a more principled discretization of the continuous Laplace-Beltrami operator, leading to more accurate curvature estimates, especially in regions of high curvature or irregular triangulation.

1.5 Modal Analysis and Spectral Mesh Reconstruction

Spectral mesh processing uses the eigendecomposition of the Laplace-Beltrami operator to analyze and process meshes. Similar to how Fourier transforms decompose signals into frequency components, the



(a) Uniform Laplacian for Plane model (b) Cotangent Laplacian for Plane model

Figure 8: Plane model Visual comparison of mean curvature estimation using uniform and cotangent Laplacian operators.

eigenvectors of the Laplace-Beltrami operator form a basis for functions defined on the mesh.

1.5.1 Spectral Decomposition

The Laplace-Beltrami operator L can be decomposed into eigenvalues λ_i and eigenvectors $\mathbf{e}_i$ such that:

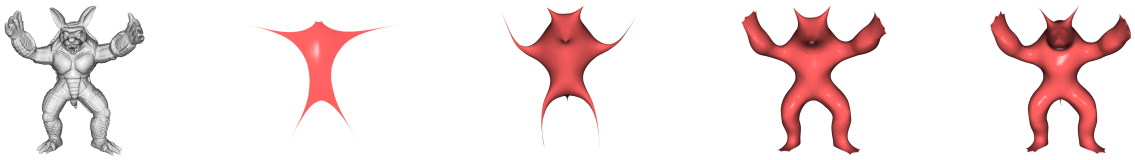
$$L\mathbf{e}_i = \lambda_i \mathbf{e}_i \tag{25}$$

The eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ represent frequencies, with smaller eigenvalues corresponding to lower-frequency, smoother functions on the mesh.

For mesh reconstruction, we project the vertex coordinates onto the eigenvectors and then reconstruct using the k smallest eigenvectors:

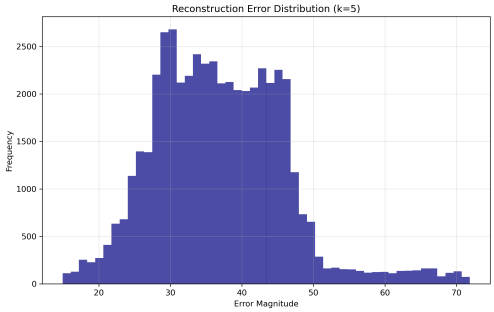
$$\mathbf{x} = \sum_{i=1}^{k} (\mathbf{x}^T \mathbf{e}_i)\mathbf{e}_i \tag{26}$$

where $\mathbf{x}$ represents the vector of x-coordinates for all vertices (and similarly for y and z coordinates).

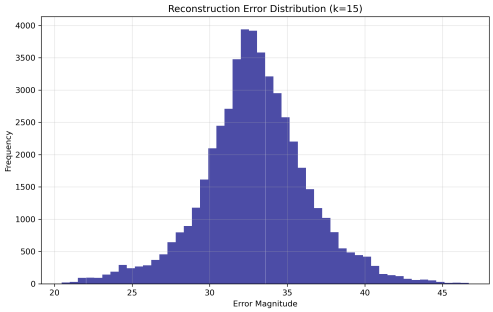


(a) Original (b) $k = 5$ (c) $k = 15$ (d) $k = 50$ (e) $k = 100$

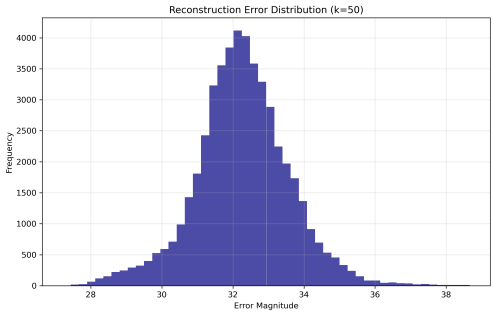
Figure 9: Spectral reconstruction of the Armadillo mesh using different numbers of eigenvectors, showing progressive improvement in detail recovery with increasing k.



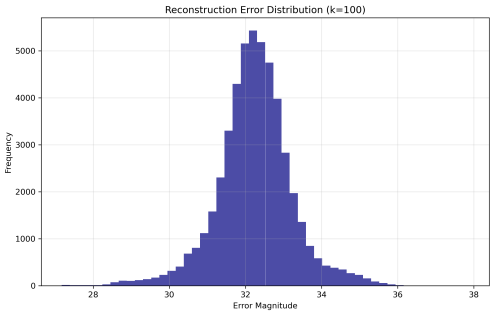
(a) Error distribution for $k = 5$



(b) Error distribution for $k = 15$



(c) Error distribution for $k = 50$



(d) Error distribution for $k = 100$

Figure 10: Distribution of per-vertex reconstruction errors for different values of k. As k increases, the error distribution becomes narrower and shifts toward smaller values, indicating more accurate reconstruction.

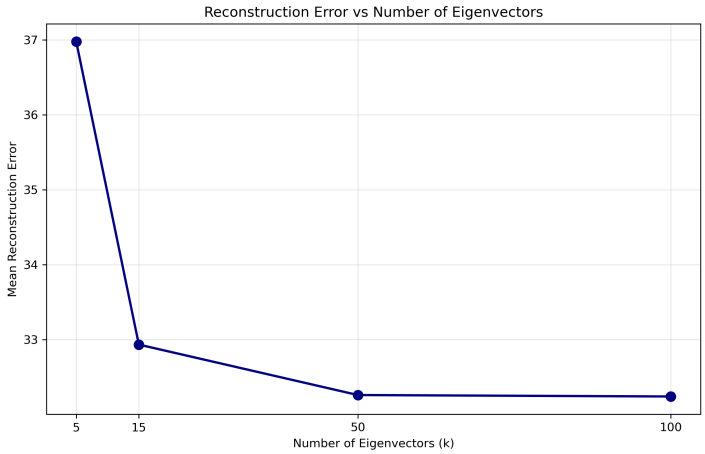


Figure 11: Reconstruction error vs. number of eigenvectors for the Armadillo model, showing diminishing returns with increasing k.

1.5.2 Reconstruction Quality Analysis

We performed spectral reconstruction on the Armadillo mesh (consisting of 49,990 vertices and 99,976 faces) using different values of k (5, 15, 50, and 100). As shown in Figure 9, the reconstruction quality improves as we increase the number of eigenvectors used:

- **Small k (e.g., $k = 5$):** With only five eigenvectors, the reconstruction captures solely the coarsest geometric features. The mesh appears overly smooth and lacks detail, with only the broad shape and major protrusions visible. The reconstruction error at this level is quite high (36.98).
- **Moderate k (e.g., $k = 15$):** With 15 eigenvectors, more geometric details emerge. The overall structure becomes better defined, though fine details are still missing. The reconstruction error decreases significantly to 32.93, representing an approximately 11% improvement over using just 5 eigenvectors.
- **Higher k (e.g., $k = 50$):** With 50 eigenvectors, we observe further improvement in geometric accuracy. Most medium-scale features are now well-represented, and the overall shape more closely approximates the original. However, the error reduction is more modest (32.26), demonstrating diminishing returns with additional eigenvectors.
- **Large k (e.g., $k = 100$):** The reconstruction with 100 eigenvectors provides only marginal improvements over $k = 50$, with the error slightly decreasing to 32.24. This minimal improvement (approximately 0.06%) despite doubling the number of eigenvectors clearly illustrates the diminishing returns of higher-frequency components.

Our eigenvalue analysis reveals interesting patterns as well. The magnitude of eigenvalues increases with k (from -0.0006 for the smallest non-zero eigenvalue at $k = 5$ to -0.034 at $k = 100$), indicating that higher eigenvectors correspond to higher-frequency components of the mesh's geometry. The computational cost also increases dramatically with k, with the eigendecomposition time rising from 388 seconds for $k = 5$ to 5,771 seconds for $k = 100$ on our hardware.

The reconstruction error, measured as the mean Euclidean distance between original and reconstructed vertices, shows a classic diminishing returns pattern. The most significant reduction in error (approximately 11%) occurs when increasing from 5 to 15 eigenvectors, while the improvement from 50 to 100 eigenvectors is negligible despite the substantially increased computational cost.

This behavior aligns with theoretical expectations and the findings of [2], confirming that the low-frequency eigenvectors of the Laplace-Beltrami operator capture the fundamental structure of the shape, while higher-frequency components contribute progressively less significant details.

For practical applications, our results suggest that $k = 15$ to $k = 50$ represents a good balance between reconstruction quality and computational efficiency for the Armadillo model. For mesh compression or coarse shape matching where broad structural features are sufficient, a smaller k (around 15) might be adequate. For applications requiring more detailed reconstruction, $k = 50$ provides good results without the substantial computational overhead of larger values.

2 Laplacian Mesh Smoothing

Laplacian mesh smoothing is a fundamental technique in geometry processing for noise reduction and surface regularization. This section explores explicit and implicit methods for mesh smoothing, analyzing their behavior, stability, and performance on various models.

2.1 Explicit Laplacian Mesh Smoothing

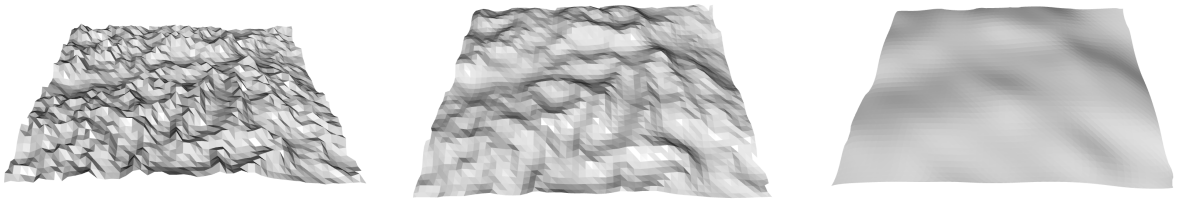
Explicit Laplacian smoothing, also known as "Diffusion Flow on Meshes," iteratively moves each vertex in the direction of the Laplacian operator applied to the vertex positions. The method follows the update rule:

$$\mathbf{x}' = \mathbf{x} + \lambda L\mathbf{x} \tag{27}$$

where $\mathbf{x}$ represents the vertex positions, L is the uniform Laplacian matrix, and λ is the step size parameter controlling the smoothing intensity.

2.1.1 Implementation and Parameter Analysis

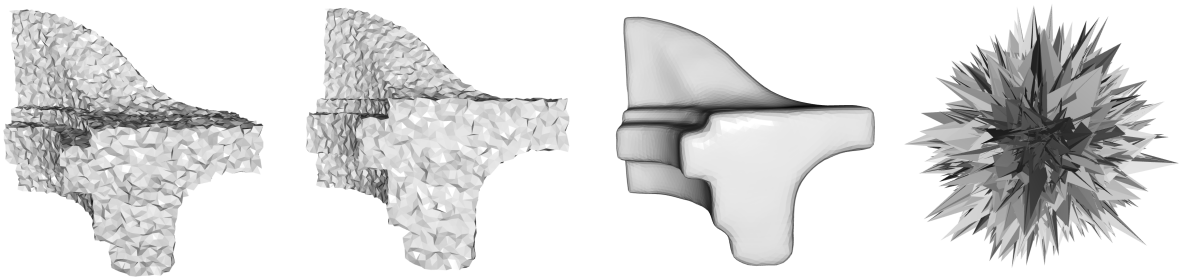
We implemented explicit Laplacian smoothing using the uniform Laplacian matrix, which assigns equal weights to all neighboring vertices. The implementation efficiently handles general triangle meshes through sparse matrix operations, making it suitable for meshes of various sizes and complexities.



(a) Original noisy mesh (b) $\lambda = 0.1$, 10 iterations (c) $\lambda = 1.0$, 10 iterations

Figure 12: Explicit Laplacian smoothing on the noisy plane mesh with different step size parameters.

The step size parameter λ plays a critical role in controlling the behavior of explicit Laplacian smoothing, as illustrated in Figure 12. Our experiments with different parameter values revealed several key characteristics:



(a) Original (b) $\lambda = 0.01$ (c) $\lambda = 0.5$ (d) $\lambda = 2.0$

Figure 13: Effect of varying step size λ on explicit Laplacian smoothing of the fandisk model (10 iterations).

Small λ Values (0.01-0.1): With small step sizes, the smoothing process is very gradual. Each iteration makes minimal changes to the vertex positions, preserving most geometric features while gently reducing noise. This conservative approach requires more iterations to achieve significant smoothing but is less likely to cause distortions or instabilities.

Moderate λ Values (0.1-0.5): For moderate values, the smoothing strikes a balanced approach between noise reduction and feature preservation. This range is typically optimal for most applications, providing effective smoothing without excessive shrinkage or distortion.

Large λ Values (1.0 and above): When using large step sizes, the smoothing becomes aggressive and can lead to several problems:

- **Mesh Instability:** Values of λ approaching or exceeding 1.0 can cause the smoothing process to become unstable, with vertices oscillating between positions or diverging rapidly.
- **Volume Shrinkage:** High λ values accelerate the shrinking effect inherent in Laplacian smoothing, causing the model to lose volume significantly after several iterations.
- **Feature Loss:** Sharp features and fine details are quickly eliminated with large step sizes, resulting in over-smoothed surfaces that lose important geometric characteristics.
- **Mesh Collapse:** In extreme cases ($\lambda > 2.0$), the mesh can collapse after multiple iterations, with vertices clustering inappropriately or the entire structure becoming severely distorted.

Figure 14 demonstrates the effect of iteration count on the smoothing process, showing how increasing iterations gradually removes noise but can also lead to feature loss and shrinkage.

(a) 1 iteration (b) 5 iterations (c) 10 iterations

Figure 14: Effect of iteration count on explicit Laplacian smoothing of the fandisk model with $\lambda = 0.1$.

Based on our experiments, we conclude that a good step size for explicit Laplacian smoothing typically falls in the range of 0.1-0.3, with the optimal value depending on the specific mesh characteristics and the desired level of smoothing. If the step size is too large (above 0.5), the smoothing process can become unstable, resulting in excessive volume shrinkage and loss of important features.

This behavior aligns with the mathematical analysis provided by [3], who demonstrated that the stability condition for explicit Laplacian smoothing is tied to the eigenvalues of the Laplacian matrix. For the uniform Laplacian, the theoretical stability threshold is $\lambda < 1/\lambda_{max}$, where λ_{max} is the largest eigenvalue of the Laplacian.

2.2 Implicit Laplacian Mesh Smoothing

Implicit Laplacian smoothing addresses the limitations of explicit smoothing by employing an implicit time integration scheme. The key difference lies in the formulation of the vertex update rule:

$$(\mathbf{I} - \lambda\mathbf{L})\mathbf{x}' = \mathbf{x} \tag{28}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{L}$ is the uniform Laplacian matrix, λ is the step size parameter, and $\mathbf{x}$ and $\mathbf{x}'$ represent the current and updated vertex positions.

This approach requires solving a sparse linear system for each coordinate dimension. We implemented this using the `spsolve` function from SciPy's sparse linear algebra module to efficiently compute the solution:

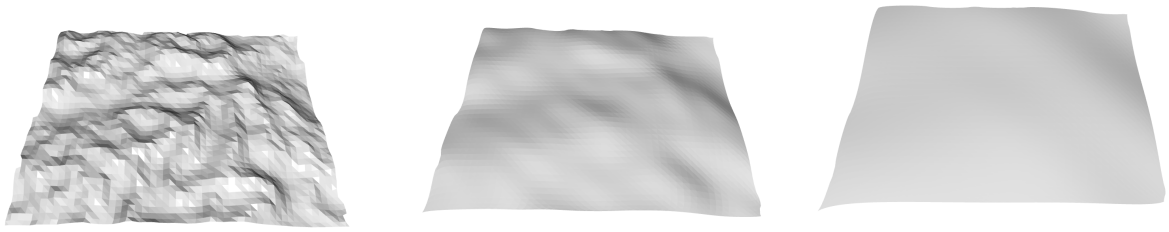
$$\mathbf{x}' = (\mathbf{I} - \lambda\mathbf{L})^{-1}\mathbf{x} \tag{29}$$

2.2.1 Parameter Analysis and Stability

The behavior of implicit Laplacian smoothing varies significantly with different λ values, as Figure 15:

Small λ Values (0.01-0.1): With small step sizes, implicit smoothing produces results similar to explicit smoothing but with slightly better stability. The smoothing effect is minimal and gradual, requiring many iterations for significant noise reduction.

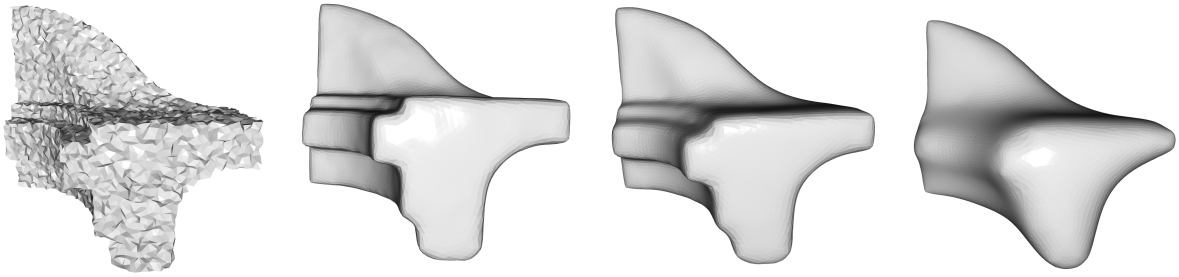
Moderate λ Values (0.1-1.0): In this range, implicit smoothing demonstrates its advantages. At $\lambda = 0.5$, the noise is effectively removed while preserving important geometric features better than the explicit method with the same parameters. The fandisk model, in particular, maintains its sharp edges more faithfully.



(a) $\lambda = 0.1$, 10 iterations (b) $\lambda = 1.0$, 10 iterations (c) $\lambda = 5.0$, 10 iterations

Figure 15: Implicit Laplacian smoothing on the noisy plane mesh with different step size parameters.

Large λ Values (1.0-5.0): This is where implicit smoothing truly distinguishes itself from explicit methods. At $\lambda = 5.0$, where explicit smoothing would have become unstable, the implicit method continues to produce reasonable results. The smoothing is more aggressive but remains controlled, with less volume shrinkage than would occur with explicit smoothing at much smaller λ values.



(a) Original (b) $\lambda = 0.5$ (c) $\lambda = 1.0$ (d) $\lambda = 5.0$

Figure 16: Effect of varying step size λ on implicit Laplacian smoothing of the fandisk model (10 iterations).

The mathematical reason for this improved stability can be understood by examining the eigenvalue response of the implicit system. For a step size λ, the amplification factor for the eigenmode corresponding to eigenvalue μ of the Laplacian is:

$$g(\mu) = \frac{1}{1 + \lambda\mu} \tag{30}$$

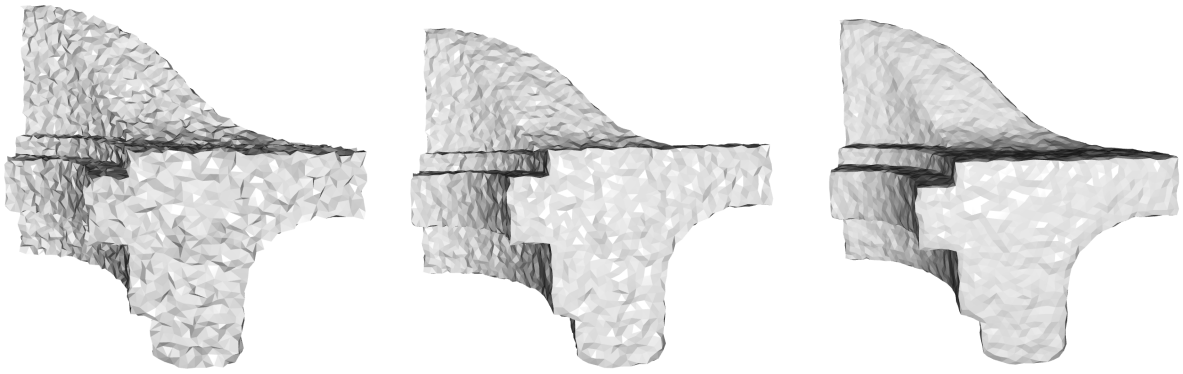
This function is bounded by 1 for all positive μ and any positive λ, ensuring numerical stability. This property allows implicit methods to use much larger step sizes without becoming unstable, as demonstrated by [4].

Figure 17 shows the effect of iteration count on the implicit smoothing process. The number of iterations interacts with the step size to determine the final smoothing result. At 1 iteration with $\lambda = 0.1$, minimal smoothing occurs, showing that even with implicit methods, a single iteration is rarely sufficient. At 5 iterations, significant noise reduction becomes evident, and at 10 iterations with moderate to high λ values, the results approach optimal smoothing for most applications.

2.3 Comparison of Explicit and Implicit Methods

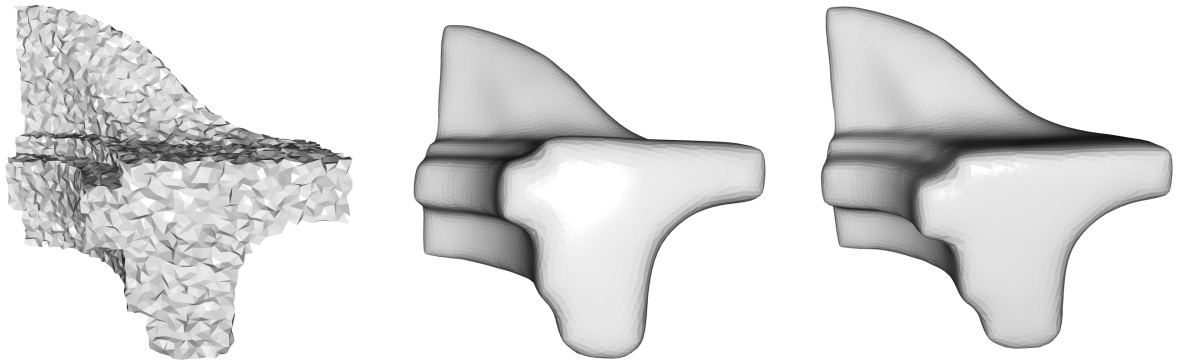
To directly compare the performance of explicit and implicit Laplacian smoothing, we applied both methods to the same models with identical parameters and analyzed the differences in quality, stability, and computational efficiency.

Key findings from our comparative analysis include:



(a) 1 iteration (b) 5 iterations (c) 10 iterations

Figure 17: Effect of iteration count on implicit Laplacian smoothing of the fandisk model with $\lambda = 0.1$.



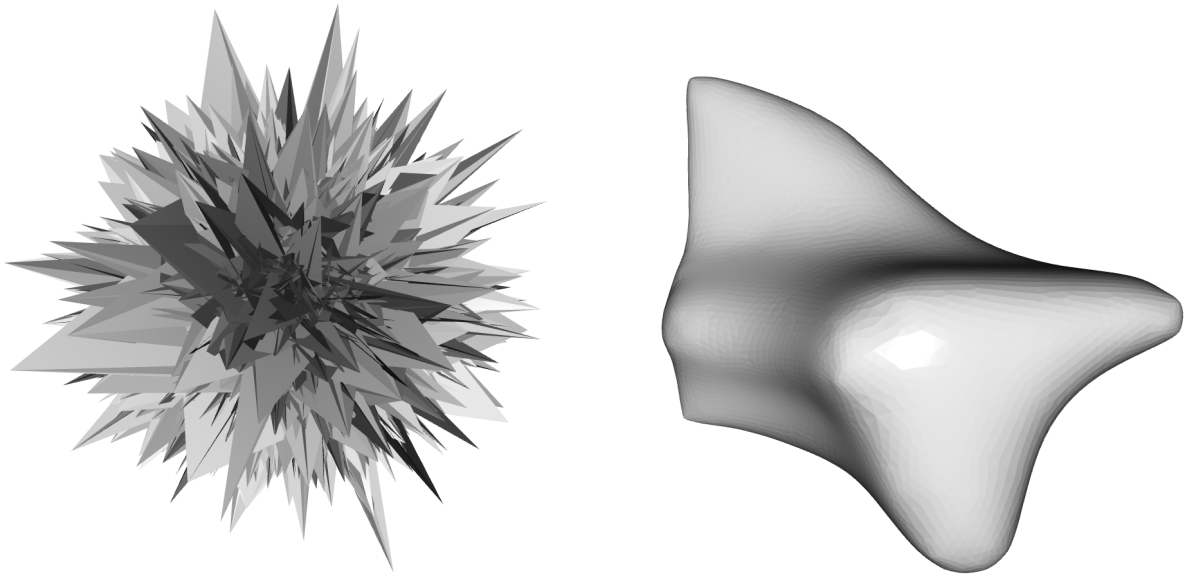
(a) Original noisy mesh (b) Explicit ($\lambda = 1.0$) (c) Implicit ($\lambda = 1.0$)

Figure 18: Comparison of explicit and implicit Laplacian smoothing on the fandisk model with identical parameters ($\lambda = 1.0$, 10 iterations).

- **Stability:** At $\lambda = 1.0$ with 10 iterations (Figure 18), the explicit method shows signs of instability and excessive smoothing, while the implicit method maintains stability and better preserves features.
- **Feature Preservation:** The implicit method consistently preserves sharp edges better, particularly visible on the corners and edges of the fandisk model.
- **Volume Preservation:** The implicit method exhibits less shrinkage, maintaining the overall volume of the models better than explicit smoothing.
- **Performance at High λ Values:** At $\lambda = 5.0$ (Figure 19), the implicit method continues to function effectively, while the explicit method produces severely distorted results.

The mathematical explanation for these differences lies in how the methods treat different frequency components of the mesh. Explicit methods can amplify high-frequency components when λ is too large, causing instability. In contrast, implicit methods naturally damp all frequency components proportionally [4].

While explicit methods are computationally simpler per iteration, the superior stability and quality of results from implicit smoothing generally justify the additional computational cost, particularly for complex models or when higher smoothing quality is required.



(a) Explicit ($\lambda = 5.0$, 10 iterations)

(b) Implicit ($\lambda = 5.0$, 10 iterations)

Figure 19: Performance comparison with high smoothing parameter ($\lambda = 5.0$) showing the superior stability of implicit methods.

2.4 Performance Evaluation of Laplacian Mesh Denoising

Mesh denoising is a critical task in geometry processing, particularly when working with scanned or acquired data. This section presents a systematic evaluation of Laplacian smoothing techniques for mesh denoising, comparing explicit and implicit methods under different noise conditions and parameter settings.

2.4.1 Experimental Setup

To evaluate the performance of Laplacian mesh denoising, we designed a comprehensive evaluation framework consisting of:

- **Controlled Noise Generation**: We added Gaussian noise to clean meshes with varying noise levels (α) proportional to the mesh's bounding box diagonal, ensuring consistency across different model scales.
- **Parameter Testing**: We tested both explicit and implicit Laplacian smoothing methods with different step sizes ($\lambda = 0.1, 0.5, 1.0$) and a fixed iteration count of 10.
- **Quantitative Metrics**: We measured several error metrics including RMSE (Root Mean Square Error), volume change percentage, and mean dihedral angle difference to assess both geometric accuracy and feature preservation.

The evaluation was performed on the fandisk model, which contains both smooth regions and sharp features, making it suitable for assessing both overall smoothing quality and feature preservation.

2.4.2 Noise Level Impact

The effectiveness of Laplacian smoothing methods varies significantly with noise levels. Figure 21 shows how RMSE changes with increasing noise levels for both explicit and implicit methods with different λ values.

For the explicit method, the optimal λ value depends on the noise level:

- At low noise ($\alpha = 0.005$), $\lambda = 0.1$ provides the best results with RMSE of 0.020

(a) Original clean mesh (b) With added noise ($\alpha = 0.02$) (c) Denoised (explicit, $\lambda = 0.5$)

Figure 20: Illustration of the mesh denoising process on the fandisk model.

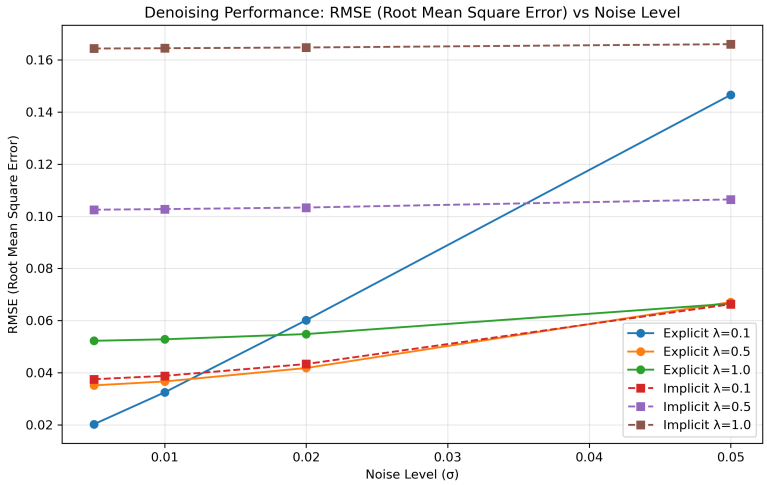


Figure 21: RMSE as a function of noise level for different denoising methods and parameters.

- At medium noise ($\alpha = 0.02$), $\lambda = 0.5$ performs best with RMSE of 0.042
- At high noise ($\alpha = 0.05$), $\lambda = 1.0$ becomes optimal with RMSE of 0.067

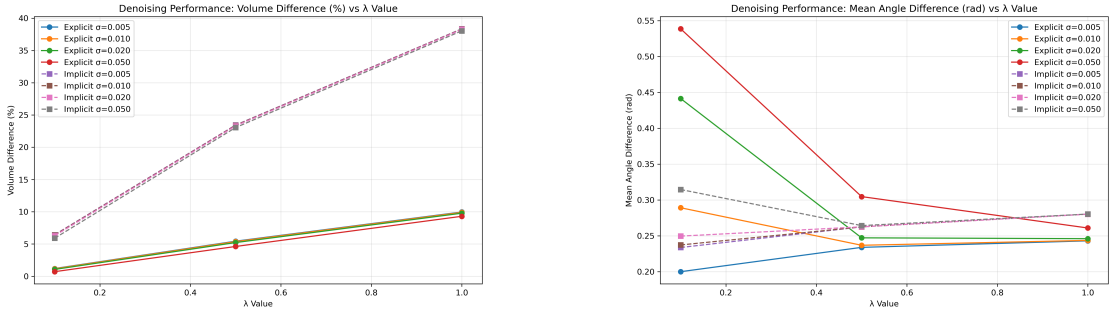
This pattern is intuitive: higher noise levels require stronger smoothing (larger λ values) to effectively remove the noise. However, stronger smoothing also increases the risk of over-smoothing, which leads to feature loss.

2.4.3 Explicit vs. Implicit Methods

A key finding from our experiments is the significant difference in behavior between explicit and implicit Laplacian smoothing methods, particularly in terms of volume preservation and feature retention.

Figure 22 reveals that:

- **Volume Preservation**: Explicit methods exhibit significantly better volume preservation. With $\lambda = 0.1$, the explicit method shows volume changes of only 0.7%-1.2%, while the implicit method reaches 5.9%-6.4%. This gap widens dramatically as λ increases, with the implicit method showing volume shrinkage of up to 38% at $\lambda = 1.0$.
- **Feature Preservation**: Both methods show increased feature smoothing (measured by mean



(a) Volume change vs. λ value

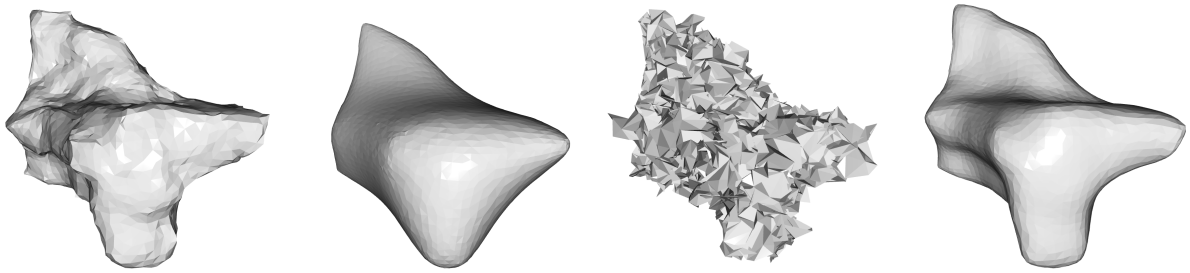
(b) Dihedral angle change vs. λ value

Figure 22: Comparison of volume preservation and feature retention between explicit and implicit methods.

dihedral angle difference) as λ increases, but the explicit method maintains more consistent behavior across different λ values. The implicit method shows a steeper increase in angle changes from $\lambda = 0.1$ to $\lambda = 0.5$.

- **Stability at High λ**: While explicit methods can become unstable at high λ values, implicit methods remain stable even at $\lambda = 1.0$, as demonstrated by the consistent error metrics across noise levels.

This difference in behavior can be understood through the mathematical formulation. As noted by Desbrun et al. [4], the implicit update rule $(I - \lambda L)\mathbf{x}' = \mathbf{x}$ provides unconditional stability, as the amplification factor for any eigenmode is bounded by 1. In contrast, the explicit update rule $\mathbf{x}' = \mathbf{x} + \lambda L\mathbf{x}$ can amplify certain frequency components when λ is too large.



(a) Explicit, $\lambda = 0.1$ (b) Explicit, $\lambda = 1.0$ (c) Implicit, $\lambda = 0.1$ (d) Implicit, $\lambda = 1.0$

Figure 23: Comparison of denoising methods on high noise ($\alpha = 0.05$) data.

2.4.4 Feature Preservation Analysis

A critical aspect of mesh denoising is the preservation of important geometric features such as edges and corners. We analyzed feature preservation by examining the distribution of dihedral angles before and after denoising, as well as through visual inspection of the results.

Figure 24 shows how different methods affect the distribution of dihedral angles. The original mesh (green) has a distinct peak at small angles, corresponding to sharp features. The noisy mesh (red) shows a more dispersed distribution. After smoothing:

- Explicit methods with moderate λ values (0.5) preserve the bimodal distribution characteristic of CAD models with sharp edges
- Explicit methods with high λ values (2.0) begin to erode sharp features

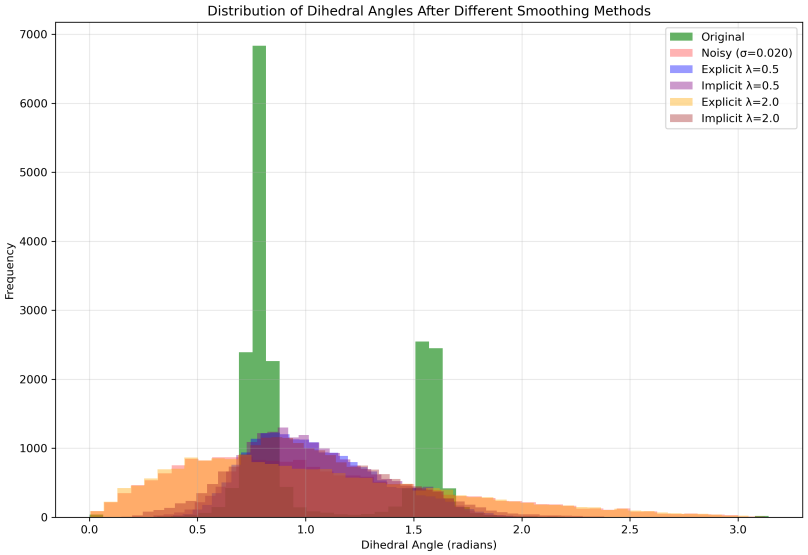
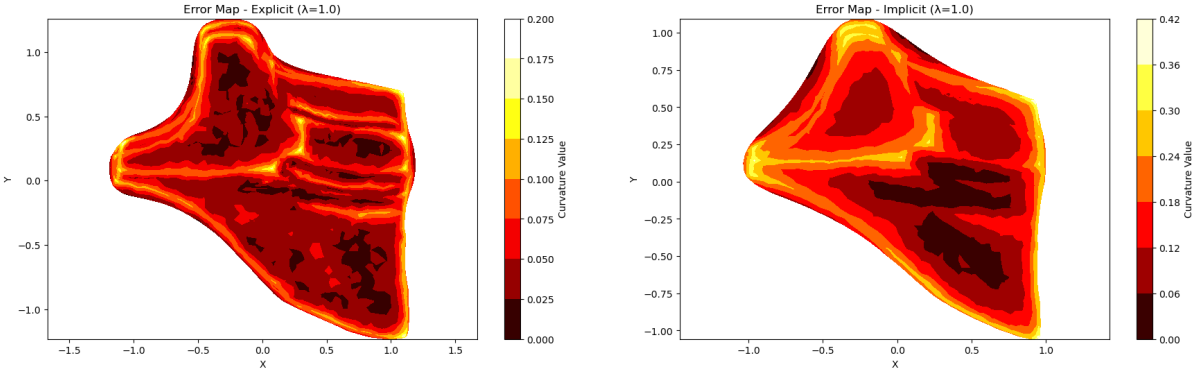


Figure 24: Distribution of dihedral angles after different smoothing methods, showing how feature sharpness is affected.

- Implicit methods, even at lower λ values, tend to shift the distribution toward medium angles, indicating feature smoothing



(a) Error map for explicit method ($\lambda = 1.0$)

(b) Error map for implicit method ($\lambda = 1.0$)

Figure 25: Error distribution maps showing where denoising introduces the most deviation from the original geometry.

Error maps in Figure 25 further illustrate where each method introduces deviations from the original geometry. The explicit method tends to concentrate errors at sharp features and corners, while the implicit method shows a more uniform error distribution but with higher overall magnitude.

2.4.5 Optimal Parameter Selection

Based on our comprehensive evaluation, we can provide recommendations for optimal parameter selection depending on the noise level and specific requirements:

For most practical applications, explicit Laplacian smoothing with λ in the range of 0.1-0.5 (depending on noise level) provides the best balance between denoising effectiveness and feature preservation. The implicit method should be reserved for cases where stability is paramount, and even then, with small λ

Noise Level	Method	λ Value	Primary Benefit
Low ($\alpha < 0.01$)	Explicit	0.1	Minimal distortion
Medium ($0.01 \leq \alpha \leq 0.03$)	Explicit	0.5	Best noise/feature balance
High ($\alpha > 0.03$)	Explicit	1.0	Better noise removal
Any (with sharp features)	Explicit	0.1-0.5	Feature preservation
Any (requiring stability)	Implicit	0.1	Numerical stability

Table 1: Recommended parameters for Laplacian mesh denoising.

values to minimize volume shrinkage.

2.4.6 Limitations and Failure Cases

Laplacian mesh denoising shows clear limitations in certain scenarios:

- **High-frequency noise**: Both methods struggle with very high-frequency noise, as shown in the results for $\alpha = 0.05$
- **Feature blurring**: Even with optimal parameters, some blurring of sharp features is inevitable
- **Volume shrinkage**: Especially for implicit methods, significant volume loss can occur
- **Parameter sensitivity**: Results are highly dependent on proper parameter selection

For extreme noise cases or when feature preservation is critical, more sophisticated approaches such as bilateral filtering [5] or normal-based filtering [6] may be more appropriate, as they can better distinguish between noise and actual geometric features.

3 Conclusion

This report has presented a comprehensive exploration of various aspects of Laplacian operators in geometry processing, focusing on mesh curvature analysis, spectral methods, and mesh smoothing.

3.1 Summary of Findings

Our investigation reveals several key insights into geometry processing techniques:

- **Curvature Estimation**: We demonstrated how the choice of Laplacian operator significantly affects curvature estimation quality. The cotangent-weighted Laplace-Beltrami operator provides more accurate curvature estimates than the uniform Laplacian, particularly on complex geometries.
- **Fundamental Forms**: Through analytical derivation of the helicoid surface's first and second fundamental forms, we confirmed the relationship between surface parameterization, normal curvature, and local geometry classification.
- **Spectral Mesh Processing**: Our experiments with the Armadillo model showed that spectral mesh reconstruction with just 15-50 eigenvectors can capture the essential geometric features, with diminishing returns for higher eigenvector counts.
- **Laplacian Smoothing**: We compared explicit and implicit Laplacian smoothing methods, finding that explicit methods offer better volume and feature preservation, while implicit methods provide superior stability at higher smoothing levels.
- **Denoising Performance**: Our systematic evaluation demonstrated that the optimal smoothing approach depends on noise level, mesh characteristics, and specific requirements for feature preservation versus noise removal.

3.2 Practical Implications

The findings in this report have several practical implications for geometry processing applications:

- For accurate curvature estimation, the cotangent Laplacian should be preferred over the uniform Laplacian, especially for meshes with irregular triangulation.
- Spectral mesh processing provides an efficient means of filtering and representing meshes at multiple scales, with applications in compression, analysis, and shape matching.
- For mesh smoothing and denoising applications, explicit Laplacian methods with moderate step sizes ($\lambda = 0.1 - 0.5$) provide the best balance between noise reduction and feature preservation in most scenarios.
- When computational stability is critical, implicit methods can be used, but with careful parameter selection to minimize volume shrinkage.

In conclusion, Laplacian operators form a fundamental tool in geometry processing, with applications ranging from differential geometry analysis to practical mesh improvement techniques. Understanding their behavior, limitations, and optimal usage is crucial for effective 3D geometry processing in various applications.

References

[1] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and mathematics III*, pages 35–57, 2003.

[2] Bruno Lévy. Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. *IEEE International Conference on Shape Modeling and Applications*, pages 13–13, 2006.

[3] Gabriel Taubin. A signal processing approach to fair surface design. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, 1995.

[4] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, 1999.

[5] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. *ACM transactions on graphics (TOG)*, 22(3):950–953, 2003.

[6] Wangyu Zhang, Bailin Deng, Juyong Zhang, Sofien Bouaziz, and Ligang Liu. Guided mesh normal filtering. *Computer Graphics Forum*, 34(7):23–34, 2015.