Geometry Processing (GPR) - Final Laboratory Report

Michele Bonomi
https://github.com/ohhmeco/GeometryProcessing

January 2026

Note

This document is intended to be also useful for my study sessions for the exams. So if details about definitions and what things are appear, it's because of this reason. Code of Lab 1 and Lab 2 have been delivered together.

Also, I'm running the code on a very old PC (Thinkpad X200) that made me have to run the labs through the following script, which is always called script.sh and present in every lab directory.

```
> cat script.sh
MESA_GL_VERSION_OVERRIDE=3.3 MESA_GLSL_VERSION_OVERRIDE=330
./01-icp-base ../scans/bunny/scan0.ply  ../scans/bunny/scan1.ply
```

I don't lknow how much this fact has affected results in terms of quality.

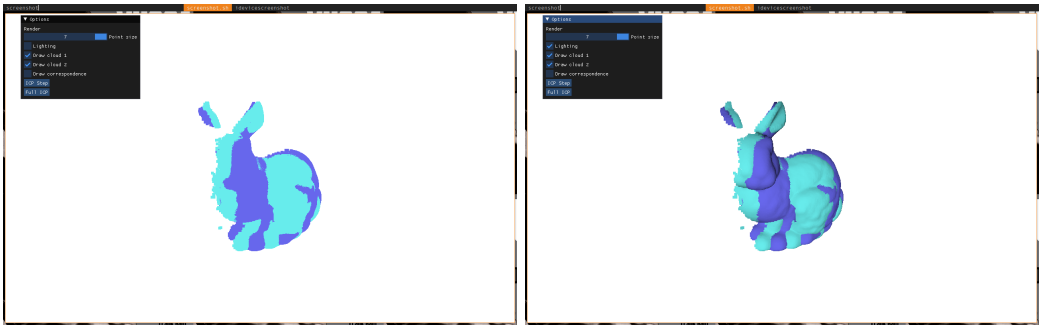
1 Lab 1: Local PCA and Feature Extraction

Status: Working as intended.

Description: Principal Component Analysis (PCA) is used to determine the principal components of a d-dimensional point set to estimate local geometry, such as surface normals.

Implementation Steps:

- Compute the centroid $\tilde{p}$ and center the points: $\tilde{p_i} = p_i - \tilde{p}$.
- Build the covariance matrix $C = \sum \tilde{p_i}\tilde{p_i}^T$.
- Compute spectral decomposition $C = V\Lambda V^T$.
- Sort eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$.
- The eigenvector v_d corresponding to the smallest eigenvalue approximates the surface normal.



Before and After

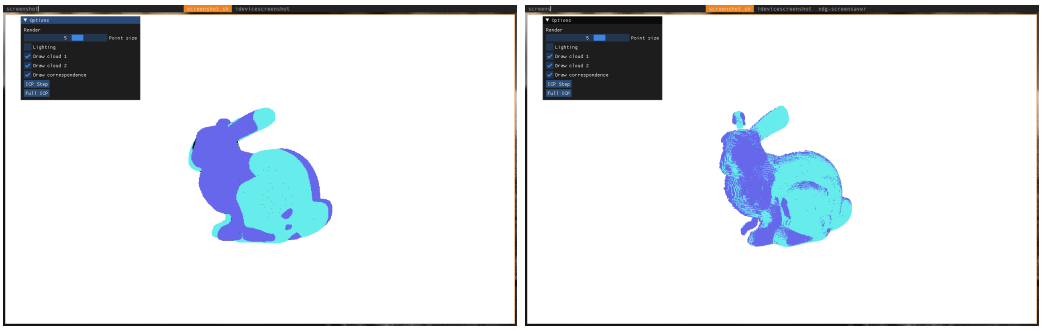
2 Lab 2: Iterative Closest Point (ICP)

Status: Working as intended.

Description: ICP aligns two point sets through an iterative process of correspondence searching and rigid transformation optimization.

Implementation Steps:

- **Border Detection:** Analyze neighbor distribution. Project neighbors onto the local XY-plane using PCA axes, compute polar angles $\alpha_j = \text{atan2}(y_j, x_j)$, and identify borders via the maximum angular gap $\Delta\alpha$.
- **Correspondence:** For each $q_i \in Q$, find the closest $p_j \in P$.
- **Optimization:** Compute $S = QP^T$ and solve via SVD ($S = U\Sigma V^T$) to find $R = VU^T$.
- **Reflection Fix:** If $\det(R) < 0$, invert the last column of the rotation matrix.
- **Transformation:** Apply $t = \tilde{p} - R\tilde{q}$ and iterate until convergence.



Before and After

3 Lab 3: Surface Reconstruction

Status: Working as intended.

Description: Generation of an implicit function $f(p)$ to represent a surface from unorganized points.

Implementation Steps:

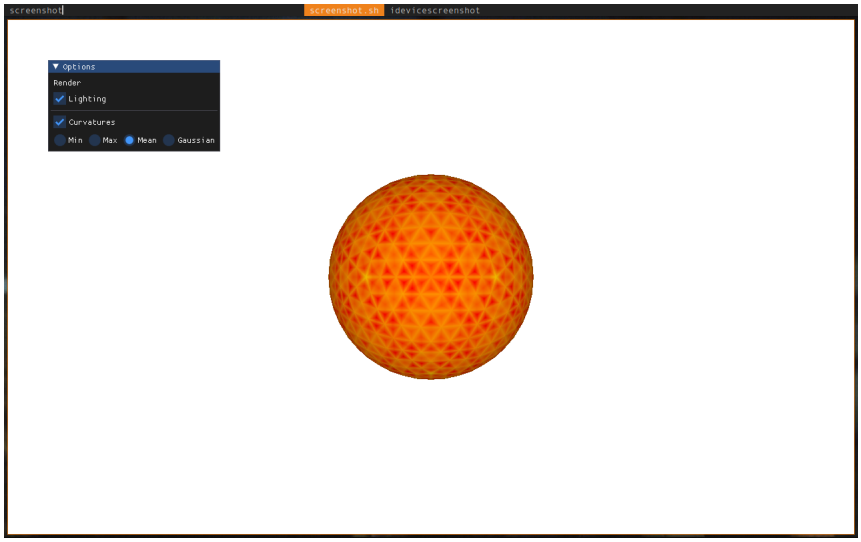
- **Hoppe et al. Approach:** Define $f(p)$ as the signed distance to the tangent plane of the closest point p_i.
- **RBF Reconstruction:** Solve the system $Ac = v$ using a Radial Basis Function $\phi(r)$.
- **Constraints:** Create artificial points $p_i \pm d \cdot n_i$ with values $\pm d$ to define the field and avoid trivial solutions.

Before and After

4 Lab 4: Curvature Estimation

Status: Working as intended.

Implementation: Local differential properties are estimated by computing Gaussian (K), Mean (H), and principal curvatures ($\kappa_{min}, \kappa_{max}$). This is achieved by fitting a local Monge Patch to the neighborhood of each vertex.



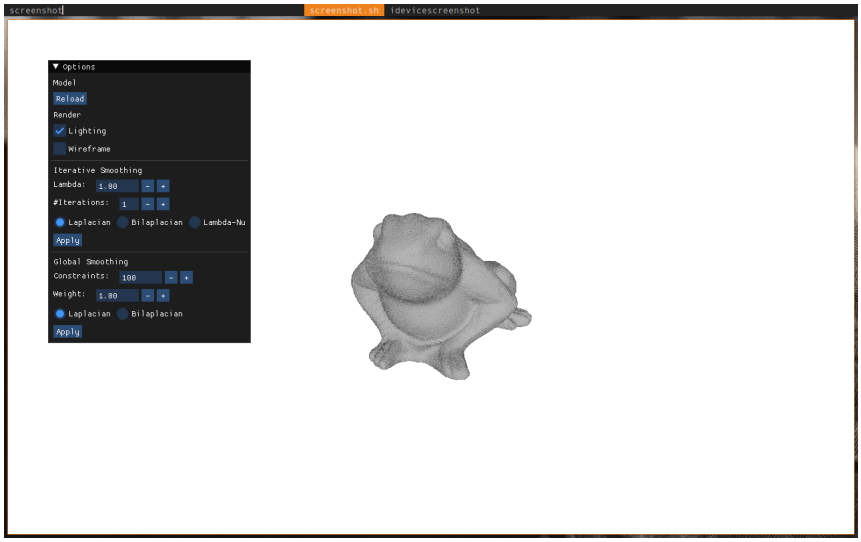
5 Lab 5: Mesh Smoothing

Status: Working as intended.

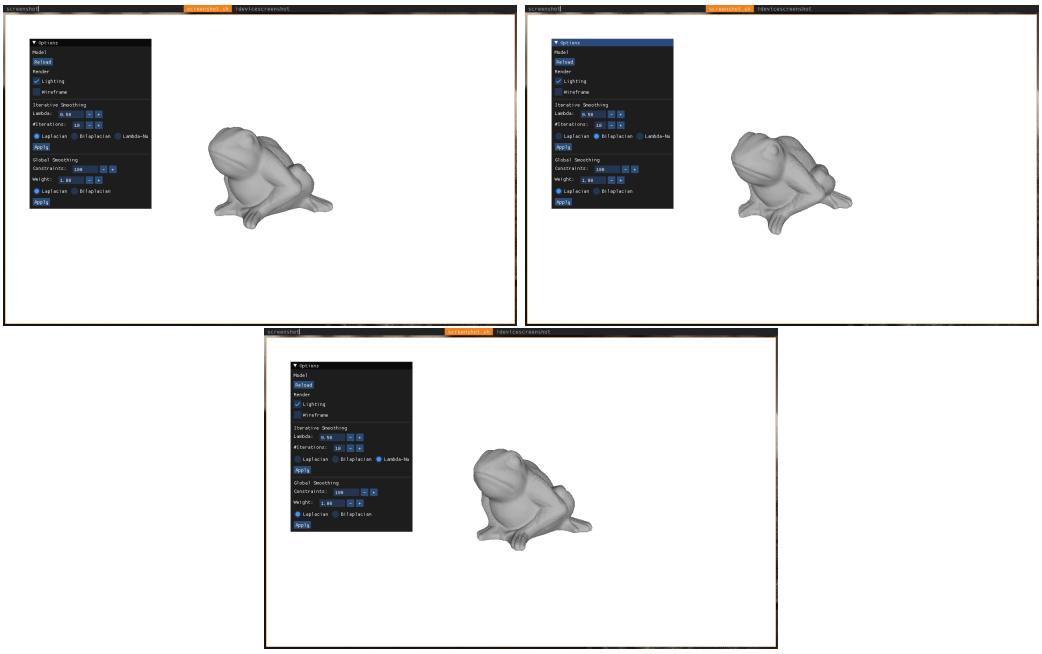
Description: Noise reduction through vertex position updates using Laplacian operators.

Implementation Steps:

- **Iterative Laplacian:** Update positions as $p_i' = p_i + \lambda\delta(p_i)$, where $\delta(p_i)$ is the average of 1-ring neighbors.
- **Taubin Smoothing:** To avoid volume loss, a two-step $\lambda - \mu$ update is applied (one positive step λ, one negative step μ).
- **Global Laplacian:** Solve $L_1 P' = b$ where unconstrained vertices satisfy the Laplacian equation (membrane energy).
- **Global Bilaplacian:** Minimize the Laplacian in a least-squares sense to preserve more geometric structure (thin plate energy).



Before



After: Laplacian, Bilaplacian, LambdaNu

6 Lab 6: Mesh Parameterization

Status: Working as intended.

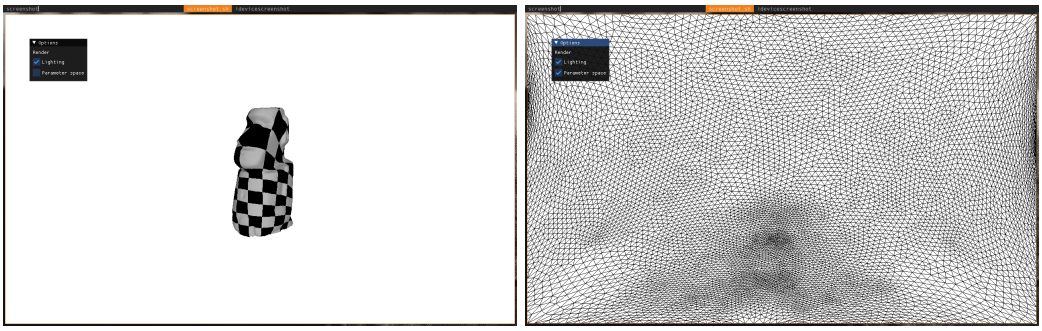
Description: Mapping a mesh with one hole to a 2D square $[0, 1] \times [0, 1]$ using Harmonic Maps.

Implementation Steps:

- **Edge Cycle:** Identify border half-edges (lacking an opposite) and chain them into a closed cycle.
- **Chord Length Mapping:** Calculate $\lambda = P/L$ for each border vertex and map to the square's perimeter:
  - If $\lambda < 0.25 \rightarrow (4\lambda, 0)$
  - Else if $\lambda < 0.5 \rightarrow (1, 4(\lambda - 0.25))$

  - Else if $\lambda < 0.75 \rightarrow (1 - 4(\lambda - 0.5), 1)$
  - Else $\rightarrow (0, 1 - 4(\lambda - 0.75))$

- **Harmonic System:** Solve the global system $LU = B$ with fixed boundary conditions and uniform Laplacian weights for internal vertices.



Before and After