Assignment 2

February 22, 2021

INSTRUCTIONS

This is the second coursework for the course COMPGV18/COMPM080 on Acquisition and Processing of 3D Geometry. The subject of this coursework is to implement and evaluate Laplacian filtering in the context of mesh denoising. The coursework is worth 100 points. You are encouraged to use C/C++, Python, or Matlab for this assignment.

LATE POLICY The coursework is due **March 29th** (Monday 23:55). We will use the departmental late submission policy. [1]

SUBMISSION GUIDELINES

- Please submit a report (PDF) on your work (to be submitted via Moodle). The report should start with a list of the questions you have attempted and to what extent you have achieved each goal.
- Next, write a short description of the methods you used for each part together with any conclusions you have drawn from your experiments.
- Color meshes and visualize result properly from at least 6 different directions.
- Copy-and-paste all your source code (excluding third-party library) into a *single* text file and name it as src.txt. Indicate the line number or the function name for each question in the report (e.g., Q1, L20, solveH).
- Include the output meshes (obj or ply format) for question 4. Name the files appropriately. (e.g., q4_arma_k10.ply).

In summary, you are expected to submit a full report (pdf), output meshes (question 4), a text file (containing all your source code), and source code files (as a zipped file along with instructions for running the code). Please do *not* upload build directories or executable files.

[1] `https://www.ucl.ac.uk/academic-manual/chapters/chapter-4-assessment-framework-taught-programmes/section-3-module-assessment#3.12`

1 CORE SECTION

In the course of handling range data or 3D scans, noise is a common source of problem. Denoising in this context refers to algorithms that attempt to remove noise, while preserving actual object features. The key challenge is to differentiate between object features and imperfections arising due to random noise. You are expected to write your own code for all the subtasks. Use third-party libraries to access neighboring vertices/edges/faces/face area is allowed (e.g. Trimesh.vertex_neighbors[2]). External functions for directly computing Laplace-Beltrami or curvature are not allowed for this coursework.

Discrete Curvature and Spectral Meshes

Given a triangle mesh, compute mean curvature (H) and Gaussian curvature (K) at each vertex. There exist several approximations to the continuous mean curvature, they differ in how they weight the influence of the neighbourhood of the vertices.

1. Uniform Laplace: Compute using the Laplace operator and uniform discretization the mean curvature H (as $\Delta\mathbf{x}/2$) at each vertex.

   Next estimate discrete Gaussian curvature K at each vertex using the angle deficit ($2\pi - \sum_j \theta_j$) at each vertex of a mesh and normalize Gaussian curvature K by area A_i. Choose a proper form of area of normalization A_i and indicate which form you use.

   Visualize mean and Gaussian curvatures separately (e.g., color code the mesh vertices). Is this a good approximation of continuous curvature? compare and discuss your results.

   (15 points)

2. First and Second Fundamental forms: Take the parameteric equations of an ellipsoid as $p(u,v) := \{a\cos(u)\sin(v), b\sin(u)\sin(v), c\cos(v\}$ for $u \in [0,2\pi)$ and $v \in [0,\pi]$. Please analytically compute the first and second fundamental forms at any point $p(u,v)$ on the ellipsoid. Use it to compute the normal curvature κ_n at the point $(a,0,0)$. (10 points)

3. Non-uniform (Discrete Laplace-Beltrami): Repeat the above computation (mean curvature H), but discretizing Laplace-Beltrami using cotangent discretization to estimate mean curvature.

   Visualize mean curvatures. Does this improve the quality of the estimated curvatures? Show your results using the input mesh: (i) curvatures/lilium_s.obj and (ii) curvatures/plane.obj. (15 points)

4. Compute and show mesh reconstructions using the smallest k eigenvectors (spectral meshes, $e_1 < e_2 < ... < e_k$) of the discrete Laplace-Beltrami operator computed above.

   Show your results using the input mesh: decompose/armadillo.obj. Set k to 5, 15, and a large value. Does the larger k give better reconstructions? Compare and discuss your results.

   Please include the output meshes in your submission. (20 points)

Laplacian Mesh Smoothing

For the following tasks, please perform your experiments on a *simple cube mesh as well as more complicated models*.

[2] `https://trimsh.org/trimesh.html?highlight=trimesh#trimesh.Trimesh.vertex_neighbors`

5. Implement *explicit Laplacian mesh smoothing* ("Diffusion Flow on Meshes" from mesh smoothing slides), and visualize before/after results.

   What is a good λ step size to use? What happens, if your step size is too large? (10 points)

6. Implement *implicit Laplacian mesh smoothing* from (Desbrun et al.: Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow, Siggraph '99, Equation (9)). Compare and visualize your results to the previous method using the input mesh: (i) smoothing/plane_ns.obj and (ii) smoothing/fandisk_ns.obj, and show the results using different step size and iterations. Compare and discuss your results. (20 points)

7. Evaluate the performance of Laplacian mesh *denoising* in the context of data smoothing. Design your tests (*e.g.*, adding increasing amounts of synthetic noise) and report your conclusions, specifically regarding when the method works and when it fails. (10 points)