Assignment 2

March 5, 2020

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.

Please submit in a short report summarizing your work (to be submitted via Moodle). The report should start with a brief list of the questions you attempted and to what extent you have achieved each item. Next, write a short description of the methods you used for each part together with any conclusions you have drawn from your experiments. *Please include result images.*

You will be required to present your work in a one to one session (date to be decided) demonstrating what you have implemented. (see question 4 and question 8 for the details).

LATE POLICY The coursework is due **March 22th** (Sunday 23:55). We will use the departmental late submission policy. [1]

[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 is allowed (e.g. igl::adjacency_list) or for python users functions mentioned in here[2]. Functions for directly computing Laplace-Beltrami includes igl::cotmatrix and igl::massmatrix are forbidden.

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. Please indicate which area normalization A_i you used.

   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. (10 points)

2. 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? (10 points)

3. 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 $k = 5, 10$, and 30. (10 points)

Laplacian Mesh Smoothing

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

4. 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? You may need to use double-precision to avoid numerical problems[3]. (15 points)

5. 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. (15 points)

[2] `https://github.com/smartgeometry-ucl/COMPM080-Tutorials-2020/tree/master/tutorials/1_coding_framework/python/4_travel`

[3] `http://eigen.tuxfamily.org/dox/group__TutorialLinearAlgebra.html`

6. 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)

7. **Demo** your implementation.
   - questions regarding your implementation

   (30 points)

Model databases: (i) *e.g.*, Dragon: `http://graphics.stanford.edu/data/3Dscanrep/`, (ii) `http://people.sc.fsu.edu/~jburkardt/data/data.html`