Geometric Data Processing (IN4255)
TU Delft, EEMCS, Department of Intelligent Systems 2023-2024 Q. 4
K. Hildebrandt Due: 20-06-2024

Practical Assignment 2

(Differential Coordinates and Shape Editing)

Task 1 (Differential Coordinates)

Implement methods that compute the sparse matrices G, which maps a function given by its function values at the vertices of a mesh to its gradient vectors, the cotangent matrix S, and the mass matrices M and M_V.

Hints:

- As a first step to compute G, compute the 3×3 matrix that maps a linear polynomial over a triangle to its gradient vector. Then use this method to compute the matrix G for a triangle mesh.
- You can use G and M to compute S.
- Design tests to check that the matrices are correctly computed.

Task 2 (Shape Editing)

Implement a simplified version of the brushes tool for editing triangle meshes we discussed in the lecture. It should allow to specify a 3×3 matrix A, which is applied to the gradient vectors of all selected faces of a triangular mesh. Then, the vertex positions of the mesh are modified such that the gradient vectors of the new mesh are as-close-as-possible (in the least-squares sense) to the modified gradients.

Remark: When constructing a mesh that best matches given gradients, the vertex positions are only determined up to translations of the whole mesh in $\mathbb{R}^3$. You can deal with this by keeping the barycenter of the mesh constant.

Task 3 (Custom Extension)

Implement one or more additional tools that use the matrices implemented. Examples are a brushes tool for smoothing surfaces using Laplace Coordinates, a shape editing tool that allows users to specify constraints on vertex positions, a mesh smoothing tool that offers implicit Laplace smoothing, a tool for hole filling in meshes, or a tool that computes geodesic distances using the heat method.

Write a report that describes and illustrates your implemented tool(s) of this task. The report should include:

- the algorithms and functionality implemented,
- how the tools can be used,

- your evaluation of parameter settings if applicable, and
- some results produced with the tools.

Record a short video (less than 1 minute), which shows the usage of your custom extension on a non-trivial mesh.

Implementation Hints

- We provide boiler-plate code for the first two tasks. The code's structure is explained in the included README, and will be discussed in more detail during the tutorial.
- For Task 1, complete the `.py` files within the `matrices` directory.
- For Task 2, complete the `.py` files within the `deformation` directory.
- For Task 3, add your custom addon within the `extension` directory (which you may rename).
- Some of the provided functions will be automatically tested. **Do not change the signatures of the provided functions or the directory's structure.**

Required deliverables on Brightspace

- For Tasks 1 and 2, complete the provided boilerplate with your implementation and upload the zipped directory.
- For task 3, upload a PDF file with your report and a video that demonstrates the functionality of your extension(s) (in MP4 format).

Deadline: Jun 20, 20:00.