Geometric Data Processing (IN4255)
TU Delft, EEMCS, Department of Intelligent Systems
K. Hildebrandt

2023-2024 Q. 4
Due: 04-06-2024

Theoretical Assignment

(Rotations, Least Squares and Laplace Matrix)

Task 1 (Rotations)

Implement a Blender add-on that extracts the rotation component of an objects transformation matrix and displays its axis and angle of rotation.

First, the 3x3 rotation matrix must be extracted from the 4x4 transformation matrix. This can be done by dropping the translation component and then normalizing to remove the scale component.

You can then compute the axis of rotation of the resulting matrix. *Hint:* one way of doing this involves the construction of a symmetric matrix and the computation of its eigenvectors.

You can also compute the angle of rotation in radians (about the aforementioned axis) from this rotation matrix. *Hint:* one way of doing this involves the trace of the rotation matrix.

Each of these steps is done in an independent function which is used by a provided blender GUI and will be tested independently. To test your own code, you should pick matrices for which you know the answer, such as the following:

$$\begin{pmatrix} \frac{1}{2}(1+\mathrm{Cos}(\sqrt{2})) & \frac{1}{2}(1-\mathrm{Cos}(\sqrt{2})) & -\frac{1}{\sqrt{2}}\mathrm{Sin}(\sqrt{2}) \\ \frac{1}{2}(1-\mathrm{Cos}(\sqrt{2})) & \frac{1}{2}(1+\mathrm{Cos}(\sqrt{2})) & \frac{1}{\sqrt{2}}\mathrm{Sin}(\sqrt{2}) \\ \frac{1}{\sqrt{2}}\mathrm{Sin}(\sqrt{2}) & -\frac{1}{\sqrt{2}}\mathrm{Sin}(\sqrt{2}) & \mathrm{Cos}(\sqrt{2}) \end{pmatrix}$$

Task 2 (Least Squares)

Implement a Blender add-on that finds the point with the minimum squared distance to a set of n planes.

A plane Q_i in three-dimensional space is defined by a point q_i and a normal vector N_i, both in $\mathbb{R}^3$. We denote by $\texttt{distance}(p, Q_i)$ the signed distance of a point $p \in \mathbb{R}^3$ to the plane Q_i. We consider the function $f(p)$ that evaluates the sum of the squared distances of a point p to all n planes,

$$f(p) = \sum_{i=1}^{n} (\texttt{distance}(p, Q_i))^2.$$

Complete the function in the provided code that finds the point p that best approximates the planes in the least squares sense, *i.e.*, that minimizes $f(p)$. The algorithm can call a function that solves a linear system.

Hint: from the lecture, we know that a quadratic polynomial on $\mathbb{R}^3$ has the form

$$p \to \frac{1}{2} p^T A p + b^T p + c,$$

for some 3×3 matrix A, vector b and constant c, and that the minimizer of a convex quadratic polynomial is the solution of the equation

$$Ap + b = 0.$$

The runtime of the implemented add-on should be independent of the number of planes. For this, implement the evaluation of the function $f(p)$ such that (after a precomputation) $f(p)$ is evaluated in constant time, in particular, independent of n.

Task 3 (Laplace Matrix and Laplace Smoothing)

Implement a method that computes the combinatorial Laplace matrix L as a sparse matrix and use it for a Blender add-on that permits smoothing a user-selected triangular mesh using explicit Laplace smoothing.

A mesh can be smoothed iteratively moving every vertex towards the average position of its neighbors using the update rule

$$p_k \leftarrow p_k - \tau \left(\frac{1}{|N_k|} \sum_{l \in N_k} p_l - p_k \right), \tag{1}$$

where p_k list the x, y, z coordinates of vertex k and N_k denotes k's neighbors. The update rule can be written using the Laplace matrix

$$\mathbf{v}_x \leftarrow \mathbf{v}_x - \tau L \mathbf{x}_x, \tag{2}$$

where $\mathbf{v}_x$ is a vector listing the x coordinates of all the vertices in the mesh. Perform the smoothing operation for the x, y, z coordinates. Use the update rule (2) in your implementation of Laplace smoothing.

Note that the number of smoothing iterations to perform (i.e., how many times to apply Eq. 2) and the weight τ are parameters the user needs to specify. Design tests to check that the computation of the Laplace matrices is correctly implemented.

Implementation Hints

- We provide boiler-plate code for the tasks in the zipped file `assignment-2.zip`. Upon decompressing this file, you can find more in-depth information about the code's structure, which functions to complete, and the test to perform in the `README.pdf` file. Also, consult the tutorials' slides (Blender onboarding and practical assignment 2) for additional information about the assignments' and code structure.

- Each task has its own directory. Add your tests in the `test.py` file within each directory. Implement unit tests with synthetic data for the functions you create to prevent failing due to improperly handled corner cases.
- The provided functions will be automatically tested, these are marked with a comment. **Do not change the signatures of these provided functions or the directory's structure.**

Required deliverables on Brightspace

- Complete the provided boilerplate code for Tasks 1, 2, and 3 with your implementation and upload the zipped directory.

Deadline: Jun 4, 20:00.