Assignment 3 (optional): (Discrete) Differential Properties and Smoothing

Handout date: 21.03.2025
Submission deadline: –
Optional, not graded

In this exercise you will

- Experiment with different ways to compute surface normals for triangle meshes.
- Reason about curvatures of smooth curves.
- Calculate curvatures from a triangle mesh.
- Implement two mesh smoothing algorithms.

1. Normals **[not graded]**

Experiment with different ways to compute vertex normals. A good starting point is the documentation inside the header file for the `per_vertex_normals()` function. Also have a look at tutorial 205 (see https://libigl.github.io/tutorial/#laplacian for explanation) for the calculation of the discrete Laplacian. Write **your own code** for computing these different normals (explicitly one-liner answers using the built-in `per_vertex_normals()` function are not allowed):

- **Standard vertex normals** These are computed as the uniform average of surrounding face normals.
- **Area-Weighted normals** Same as above, but the average of the face normals is weighted by the face area.
- **Mean-curvature normals** These are the cotangent-weighted discrete Laplacian at every vertex.
- **PCA computation** At each vertex v_i, a plane is fit to the k nearest neighbours of this vertex using Principal Component Analysis. The vertex normal is then the principal component with the smallest eigenvalue (the normal to the plane). The neighbours can be collected by running breadth-first search.
- **Normals based on quadratic fitting** Using the local frame computed at a given vertex with PCA as above, the vertex and its k-ring of neighbours (all vertices reachable from the vertex within a distance of k edges), or alternatively its k nearest neighbors, can be seen as a height function in that frame (the height axis being the principal component with the smallest eigenvalue). The derivative of that height function will then be normal to the surface. Thus the vertex normal can be found by (a) fitting a quadratic bi-variate polynomial to these height samples, (b) using the analytic expression of the polynomial derivative to compute the normal at the origin of the frame.

Relevant `libigl` *functions:* `viewer.data().set_normals()`, `igl::cotmatrix`, `igl::massmatrix`, `igl::fit_plane`, `igl::principal_curvature` (look inside for quadric fitting).

2. Smooth curvature **[not graded]**

Match the following (signed) curvature expressions (where s stands for the arc-length parameter) to the corresponding plots of curves below (you can assume the curves are plotted from $s = -\infty$ to $s = +\infty$). Briefly motivate your answer.

$$\kappa_1(s) = s^2 + 1$$
$$\kappa_2(s) = s$$
$$\kappa_3(s) = s^2$$
$$\kappa_4(s) = s^2 - 4$$

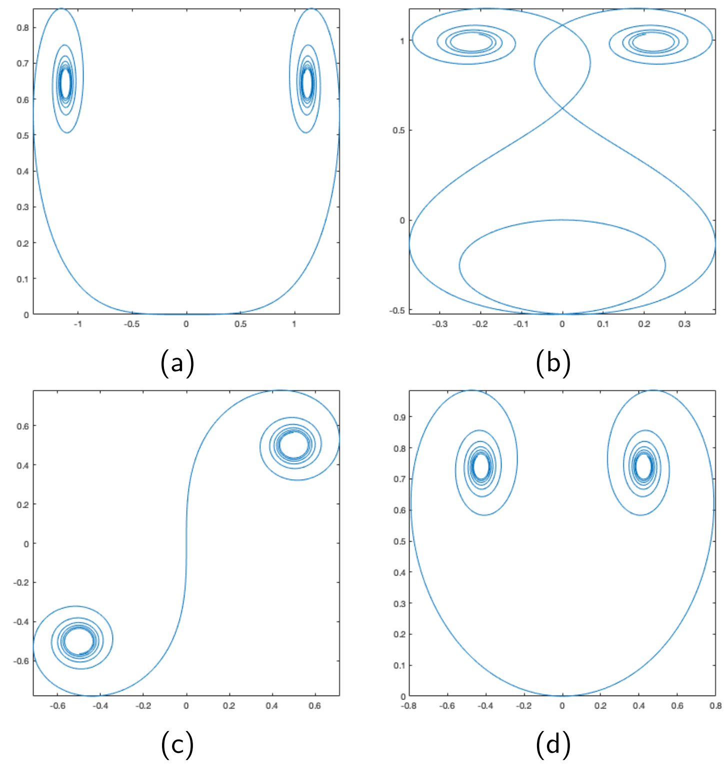
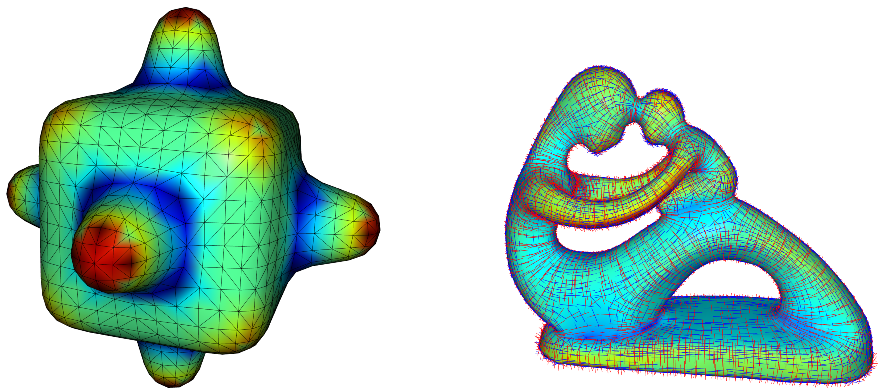


Figure 1. Match the curvature definitions $\kappa(s)$ to these curve plots.

3. Discrete curvature **[not graded]**

Compute discrete mean, Gaussian, and principal curvatures (κ_{min} and κ_{max}) using the definitions given in class. Color the mesh according to curvature by using a color map of your choice.



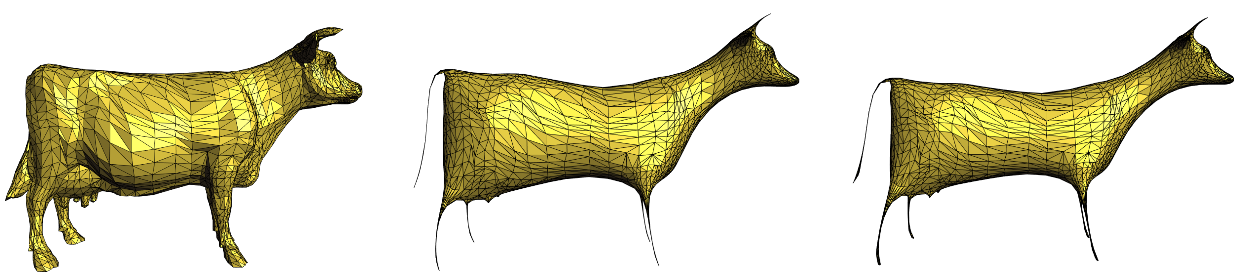
(A) Gaussian Curvature

(B) Gaussian Curvature

FIGURE 2. On the left: visualization of the Gaussian curvature. On the right, visualization of the mean curvature and principal curvature directions.

Relevant `libigl` *functions:* `igl::gaussian_curvature`, `igl::principal_curvature`, `igl::cotmatrix`, `igl::massmatrix`, `igl::jet`.

4. Smoothing with the Laplacian **[not graded]**



(A) original

(B) explicit smoothing, $\lambda = 0.01$, 1000 iterations

(C) implicit smoothing, $\lambda = 20$, 1 iteration

FIGURE 3. Explicit and implicit smoothing on the *cow* mesh.

Implement explicit Laplacian smoothing (mean curvature flow) as explained in the lecture. Experiment with uniform and cotangent weights. Have a look at tutorial 205 before you begin, where implicit smoothing has been implemented. Compare the implicit and explicit methods with each other.

Relevant `libigl` *functions:* `igl::cotmatrix`, `igl::massmatrix`, `igl::grad`, `igl::doublearea`.

5. Bilateral Smoothing **[not graded]**

Implement bilateral mesh smoothing as described in the paper "Bilateral Mesh Denoising" by Fleishman et al. [1]. Use it to smooth the noisy meshes in the data folder.

References

[1] Shachar Fleishman, Iddo Drori, and Daniel Cohen-Or. Bilateral mesh denoising. *ACM Trans. Graph.*, 22(3):950–953, July 2003.