Assignment 3: Discrete Normals, Curvatures, and Smoothing

Note: This assignment is optional and won't be graded. But next assignment assumes familiarity with differential operators (in particular, the discrete Laplacian), so we encourage you to work through it.

In this exercise you will

- Experiment with different ways to compute surface normals for triangle meshes.
- Calculate curvatures from a triangle mesh.
- Perform mesh smoothing.
- Familiarize yourselves with the relevant implementations in `libigl`.

1. Vertex Normals

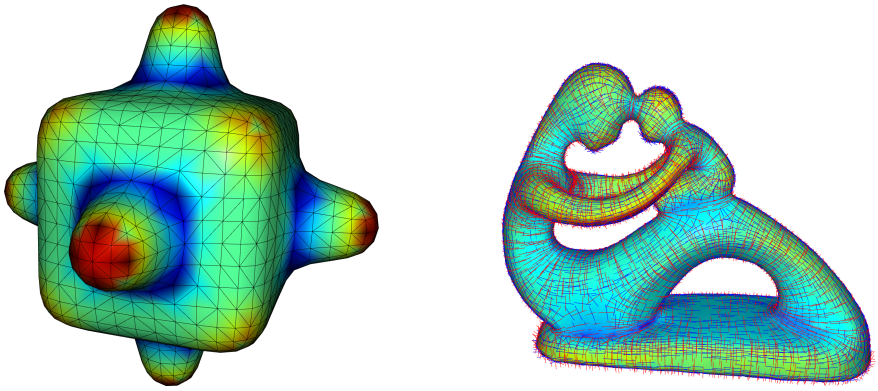
Starting from the `libigl` tutorial, experiment with different ways to compute per-vertex normals. A good starting point is tutorial # 201 and the documentation inside function `igl::per_vertex_normals`. Also check tutorial # 205 for the discrete Laplacian calculation. You will need to compute and switch between the following types of normals:

- **Standard vertex normals** These are computed as the unweighted average of surrounding face normals.
- **Area-weighted normals** Same as above, but the average is weighted by face area.
- **Mean-curvature normals** Apply the cotangent-weighted Laplacian to the mesh vertex positions to compute the normal weighted proportionally to mean curvature as shown in class. If this vector field's magnitude at a particular vertex is greater than some threshold, ϵ, use the normalized vector as the vertex's normal. Otherwise, (when the surface is locally flat or possibly saddle-shaped), compute the vertex's normal using the area-weighted average above.
- **PCA computation** At each vertex v_i, fit a plane to the k nearest neighbors (with k configurable by GUI) using Principal Component Analysis. The vertex normal is then the principal component with the smallest eigenvalue (i.e. the fit plane's normal). 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 positions of the vertex and its k neighbours can be represented as $[u, v, f(u, v)]$ using a height field function $f(u, v)$. This parametric surface is expressed in the local frame's basis (i.e. the principal component vectors), with u and v giving the tangential coordinates and $f(u, v)$ giving the height. Recall, the height axis is the principal component with smallest eigenvalue. Compute the vertex normal by (a) fitting a quadratic bivariate polynomial to the height samples f_i, then (b) computing the polynomial surface's normal at $u = v = 0$ (i.e., at the vertex) by differentiating the surface with respect to (u, v) to compute tangent vectors and then taking their cross product.

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

2. Curvature

Compute discrete mean, Gaussian, and principal curvatures (κ_{min} and κ_{max}) using the definitions from class. Color the mesh by curvature with a color map of your choice. Check out tutorials # 202, # 203 before you begin.

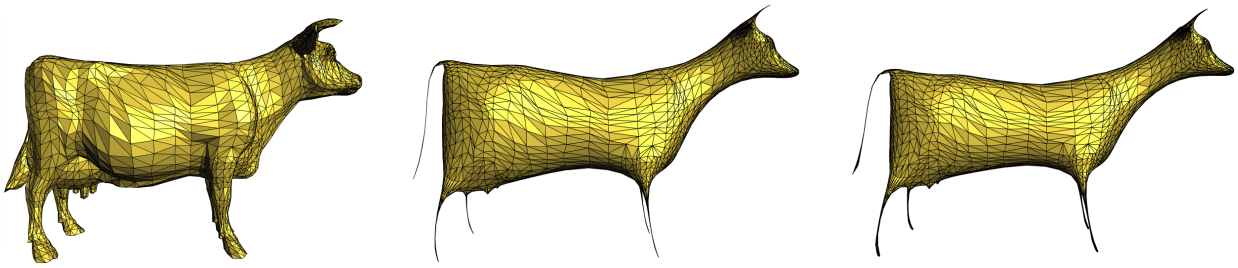


(A) Gaussian Curvature (B) Gaussian Curvature

Figure 1. On the left: Gaussian curvature visualization. On the right, mean curvature and principal curvature directions.

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

3. Smoothing with the Laplacian



(A) original

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

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

FIGURE 2. Explicit and implicit smoothing on the *cow* mesh.

Perform explicit and implicit Laplacian mesh smoothing. Experiment with uniform and cotangent weights. Before you begin, check out tutorial # 205, which implements implicit smoothing.

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

4. Bilateral Smoothing

Implement bilateral mesh smoothing as described in "Bilateral Mesh Denoising" by Fleishman et al.