Digital Geometry Processing
Exercise 4 - Surface Quality and Smoothing

Handout date: 03.06.2014
Submission deadline: 03.12.2014, 23:59

Note

Copying of code (either from other students or from external sources) is strictly prohibited! Any violation of this rule will be reported to USC Office of Student Judicial Affairs and Community Standards.

What to hand in

A .zip compressed file renamed to "Exercise*n*-YourName.zip" where *n* is the number of the current exercise sheet. It should contain:

- All the source code necessary to compile your application. Please, do not hand in libraries, build folders, binaries. You only need to submit `cc` and `hh` files.
- A "readme.txt" file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Upload the archive at `http://www.dropitto.me/usc-cs599dgp` before the deadline. Password is `ididit`.

Surface Quality and Smoothing

In this exercise you will implement the following tasks:

- Computation of uniform Laplacian approximation and using it for mean curvature approximation and smoothing (uniform Laplace smoothing)
- Computation of the Laplace-Beltrami operator and using it for mean curvature approximation and for smoothing (normal direction smoothing)
- Computation of the circumradius over minimum edge length to evaluate the triangle shapes of a mesh. Comparing the effect of the two smoothing algorithms in terms of their influence on the triangle shape.
- Approximation of the Gaussian curvature

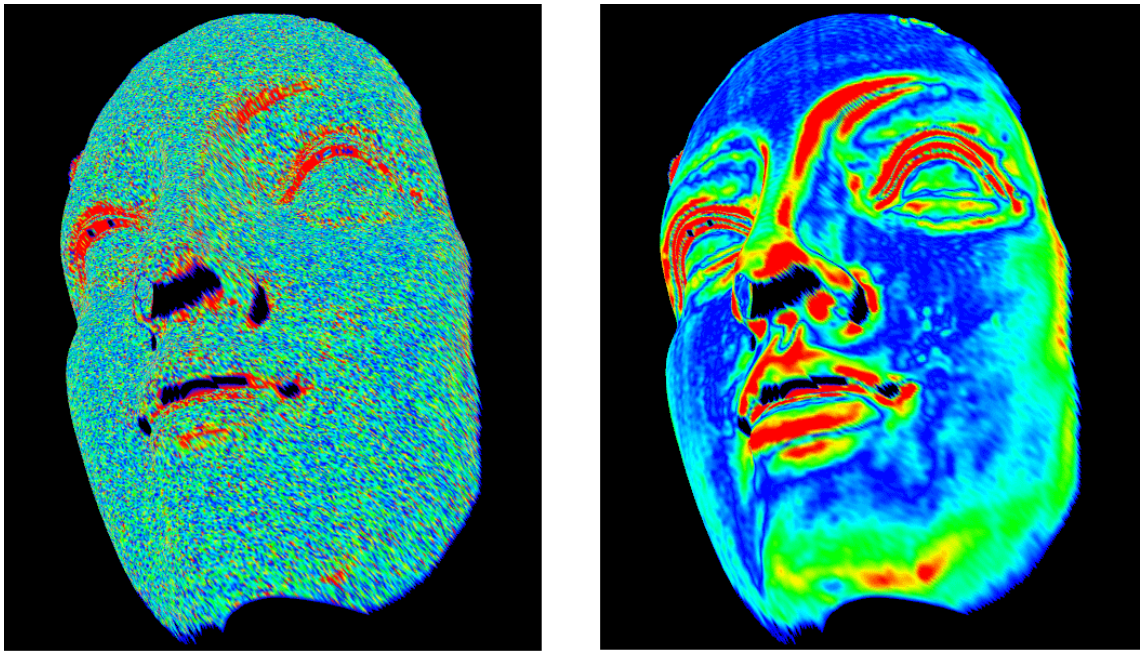


Figure 1: Mean curvature approximation on the face dataset before and after smoothing using uniform Laplace

Framework

A new project, `Smoothing` has been added to the framework from the previous exercise. It reuses files from the `ValenceViewer` project and adds additional classes. The `QualityViewer` class extends the `MeshViewer` and adds visualization modes like curvatures, triangle shapes, and reflection lines. The `SmoothingViewer` extends the `MeshViewer` and adds the smoothing operations which are triggered by the `N` and `U` keys. You will need to implement portions of the assignment in the two new classes: `QualityViewer` and the `SmoothingViewer`.

To load one of the meshes for this exercise use the following command line arguments

- to load bunny, use `data/bunny.off`
- to load face 1, use `data/max.off`
- to load face 2, use `data/scanned_face/off`

3.1 Uniform Laplace curvature and smoothing

a) The uniform Laplace operator approximates the Laplacian of the discretized surface using the centroid of the one-ring neighborhood. For a vertex v let us denote the n neighbor vertices with v_i. The uniform Laplace approximation is

$$L_U(v) = \left(\frac{1}{n}\sum_i v_i\right) - v$$

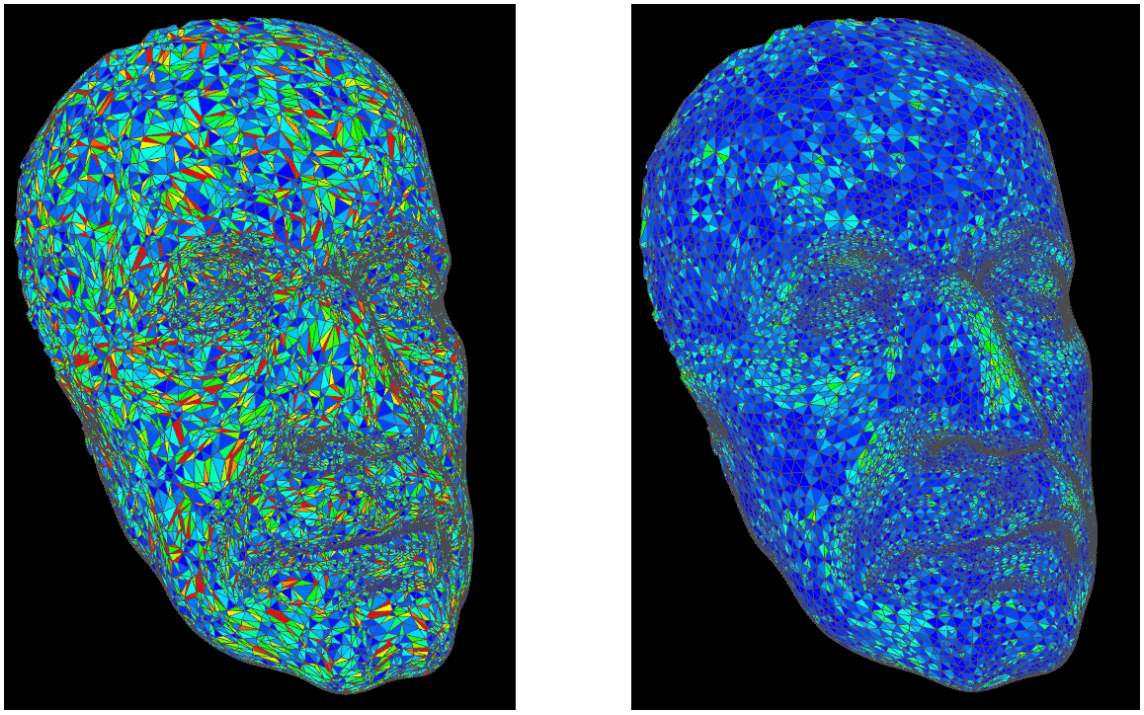


Figure 2: The shapes of the triangles before and after applying unform Laplace smoothing

The half length of the vector L_U is an approximation of the mean curvature.

Implement the `calc_uniform_mean_curvature()` function in the `QualityViewer` class. This function has to fill up the `vunicurvature_` vertex property with the mean curvature approximation using the uniform Laplace operator.

b) Implement uniform Laplace smoothing in the `uniform_smooth (unsigned int _iters)` function of the `SmoothingViewer` class. It has to apply `_iters` smoothing operations on the mesh, where one smoothing operation moves the vertices of the mesh halfway along the L_U vector:

$$v' = v + \frac{1}{2} \cdot L_U(v)$$

Hint: do not forget to update normals after vertex coordinates change.

Test your solution by loading the `scanned_face.off` model. Choose the "Uniform mean curvature" mode and apply uniform smoothing by pressing the `U` button. You should get images similar to Figure 1. You can use the "Reflection Lines" mode to see how the smoothing really changes the surface quality.

3.2 Triangle shapes

Many applications require triangle meshes with nice triangles. Equilateral triangles usually are considered "nice", skinny or flat triangles are "bad". A measure to capture this quality is the circumradius to minimum edge length ratio. The lower this ratio is, the closer the triangle is to the equilateral (ideal) triangle. To derive a formula for the circum-

radius r, one can use these two expressions for the area of a triangle:

$$A = \frac{|a| \cdot |b| \cdot |c|}{4 \cdot r} = \frac{|a \times b|}{2},$$

where a, b, and c are vectors representing the edges of the triangles, so that a and b share a common vertex as origin.

Implement the `calc_triangle_quality()` function in the `QualityViewer` class, so that it fills the face property `tshape_` with the circumradius over minimum edge length ratio for every triangle.

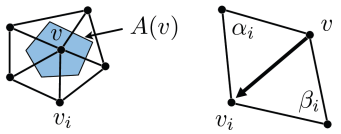
Hint: for numerical stability, make sure that if your cross product denominator is small or negative, then you simply assign a large value to the triangle shape measure and do not calculate it with the general formula.

In the visualization of triangle shapes, the scale of the color coding will be fixed between `0.6` and `2.0`, so that you can see the absolute changes induced by smoothing. Load the `max.off` model, choose the "Triangle Shapes" mode and apply uniform smoothing by pressing the `U` key. The shapes of the triangles will change as illustrated on Figure 2.

3.3 Laplace-Beltrami curvature and smoothing

For irregular meshes the uniform Laplace smoothing moves vertices not only along the surface normal, but tangentially, as well. To create a smoothing which moves vertices only along surface normals one can use the Laplace-Beltrami operator. This operator uses the certain weights for the neighbor vertices:

$$L_B(v) = \frac{1}{2A} \sum_i ((\cot \alpha_i + \cot \beta_i)(v_i - v))$$
$$L_B(v) = w \sum_i (w_i (v_i - v))$$



See the lecture slides and the above picture for explanation about this formula. Again, the half length of the Laplace approximation gives an approximation of the mean curvature.

a) Study the `calc_weights()` function to understand how and which weights are computed. Use the stored weights values to implement the mean curvature approximation using the Laplace-Beltrami operator. The `calc_mean_curvature()` function has to fill the `vcurvature_` property with the mean curvature approximation values.

b) Implement smoothing using the Laplace-Beltrami operator. In order to keep the smoothing numerically stable **use the sum of the edge weights $\sum w_i$ instead of the area weight** w to normalize the approximation:

$$L_B = \frac{1}{\sum_i w_i} \sum_i w_i (v_i - v)$$

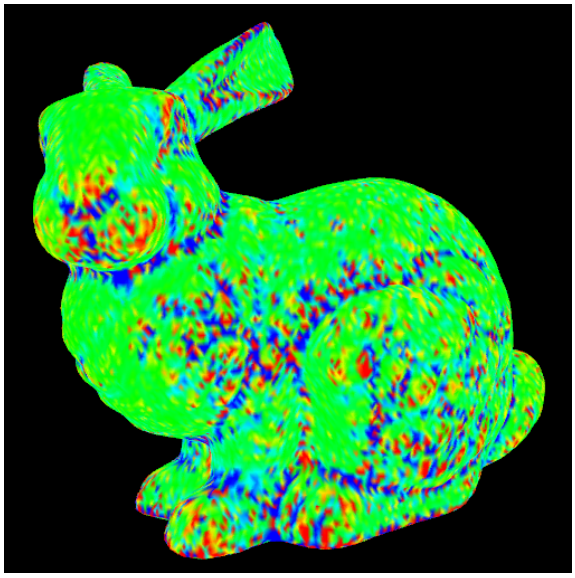


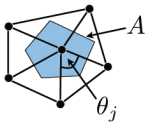
Figure 3: The Gaussian curvature approximation

Do not forget to use the damping of $\frac{1}{2}$ just like in the uniform Laplace case. Compare the effects of the two smoothing methods on the shape of the triangles using the "Triangle Shapes" visualization mode.

3.4 Gaussian curvature

In the lecture you have been presented an easy way to approximate the Gaussian curvature on a triangle mesh. The formula uses the sum of the angles around a vertex and the same associated area which is used in the Laplace-Beltrami operator:

$$G = \frac{1}{A}\left(2\pi - \sum_j \theta_j\right)$$



Implement the `calc_gauss_curvature()` function in the `QualityViewer` class so that it stores the Gaussian curvature approximations in the `vgausscurvature_` vertex property! Note that the `vweight_` property already stores $\frac{1}{2A}$ value for every vertex, you do not need to calculate A again. For the bunny dataset you should get a Gaussian curvature approximation like on Figure 3.

3.5 For the passionate (optional)

Implement the "tangential smoothing" which moves vertices only in the tangent plane of the vertex, thus focuses on enhancing triangle shapes. For this, you need to project

the uniform Laplace approximation back to the tangent plane of the vertex. Use this projection vector to compute the new position of the vertex. Notice that you need to store the original normal of the vertex additionally, in order to keep the vertices always on the original tangent plane, even after several smoothing iterations.