Surface Representation and Geometric Modeling

Exercise 2 - Surface Quality and Smoothing

Handout date:10/11/2010

Submission deadline: 10/25/2010, Midnight

Note

Copying of code (either from other students or from external sources) is strictly prohibited! Any violation of this rule will lead to expulsion from the class.

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:

- A Microsoft Visual Studio 2008 file solution with all source files and project files. In order to avoid sending build files, close your Visual Studio solution and run the file "cleanSolution.bat" located in the top directory of the framework before you submit.
- A "readme.txt" file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Send your solutions to mirela@stanford.edu before the submission deadline.
  No late submissions will be accepted.

Sample Software

See 02-Smoothing.exe for an example of how your program should run.

Surface Quality and Smoothing

In this exercise you will implement the following:

- Computation of the uniform Laplacian operator and using it for 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 circum-radius to minimum edge length ratio 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.

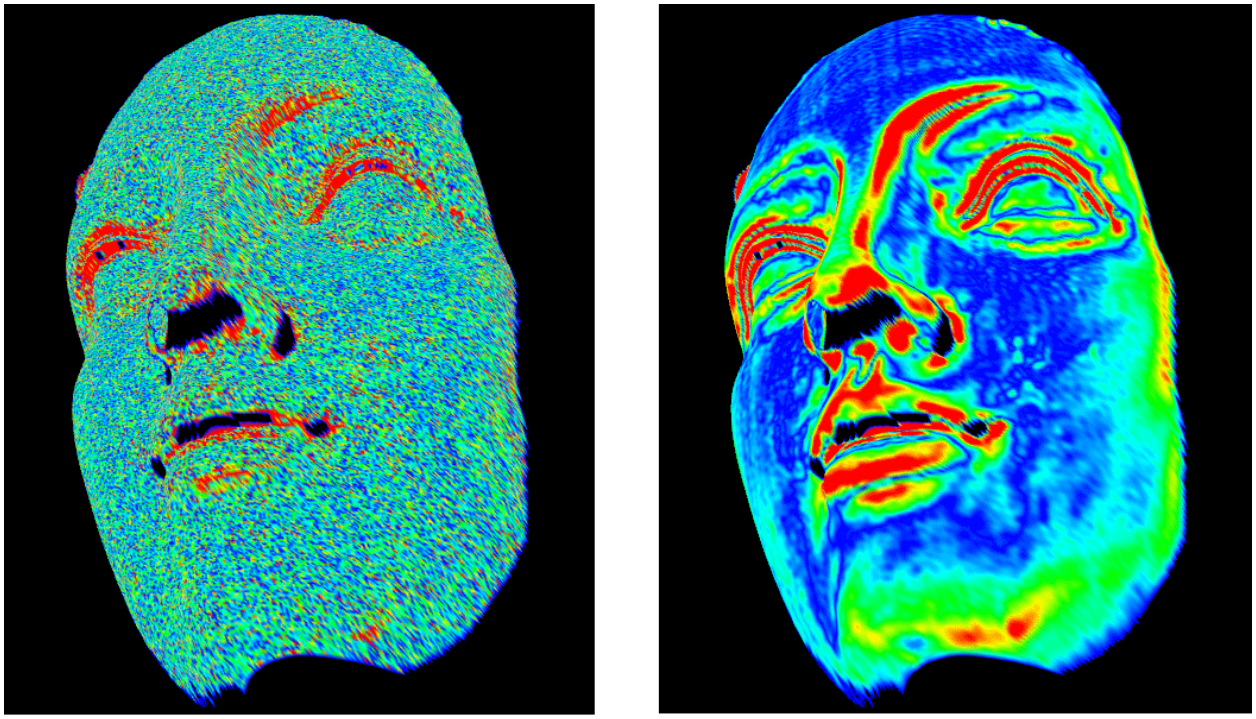


Figure 1: Mean curvature approximation on the face dataset before and after smoothing using uniform Laplace

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 in the two new classes: QualityViewer and the SmoothingViewer.

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 denote the n neighboring vertices by v_i. The uniform Laplacian approximation is:

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

The half length of the vector L_U is a (very crude) approximation of the mean curvature.

Implement the calc_uniform_mean_curvature() function in the QualityViewer class. This function should 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 should apply _iters smoothing operations on the mesh, where one smoothing operation moves each vertex of the mesh halfway along its L_U vector:

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

Hint: Make sure that all vertices are smoothed in parallel. Do not forget to update vertex 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.

Note: The smoothing operation may be not well defined for boundary vertices. Think yourself how to smooth these vertices and describe your solution in the readme file.

3.2 Triangle shapes

Many applications require triangle meshes with nicely shaped triangles. Equilateral triangles usually are considered "nice", skinny or flat triangles are "bad". A measure of this quality is the ratio between triangle circum-radius and minimum edge length. The smaller 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:

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

where a, b and c are vectors representing the edges of the triangles, such 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 this 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.

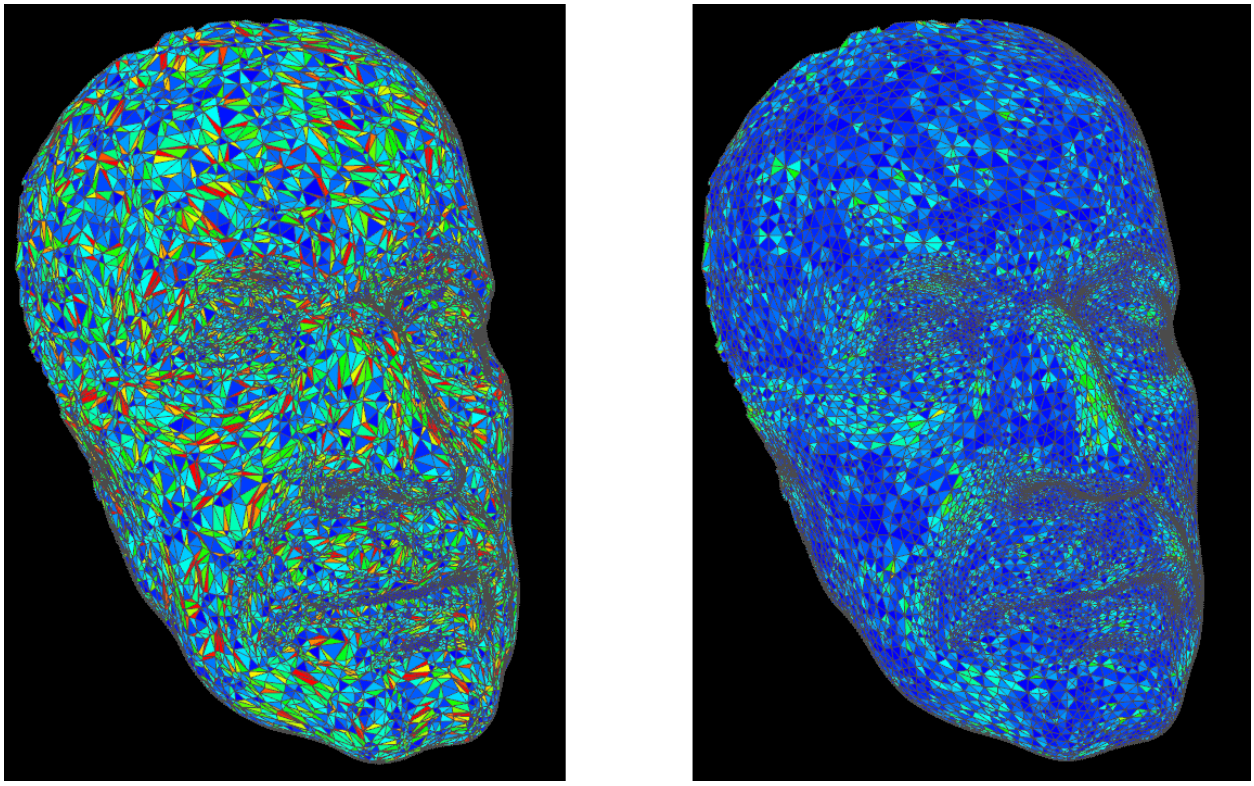


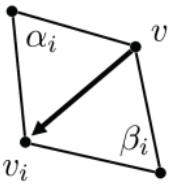
Figure 2: The shapes of the triangles before and after applying unform Laplace smoothing

3.3 Laplace-Beltrami mean curvature and smoothing

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

$$L_B(v) = \sum_i w_i(v_i - v) \quad = \sum_i (\cot \alpha_i + \cot \beta_i)(v_i - v)$$

where the two angles are those opposite the edge between v and v_i:



The Laplace-Beltrami vector gives an approximation of the normal and its half length an approximation of the mean curvature at that vertex.

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

b) Implement smoothing using the Laplace-Beltrami operator. Normalize the cotangent edge weights to sum to unity: $L_B = \frac{1}{\sum_i w_i} \sum_i w_i (v_i - v)$. Do not forget to use the damping factor of $\frac{1}{2}$ just like in the uniform smoothing 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 were shown an easy way to approximate the Gaussian curvature at a vertex of a triangle mesh. This is merely the *angle defect* - using the sum of the angles around a vertex:

$$G = 2\pi - \sum_j \theta_j$$

Implement the calc_gauss_curvature() function in the QualityViewer class so that it stores the Gaussian curvature approximations in the vgausscurvature_ vertex property. For the bunny dataset you should get a Gaussian curvature approximation similar to the one in Figure 3.

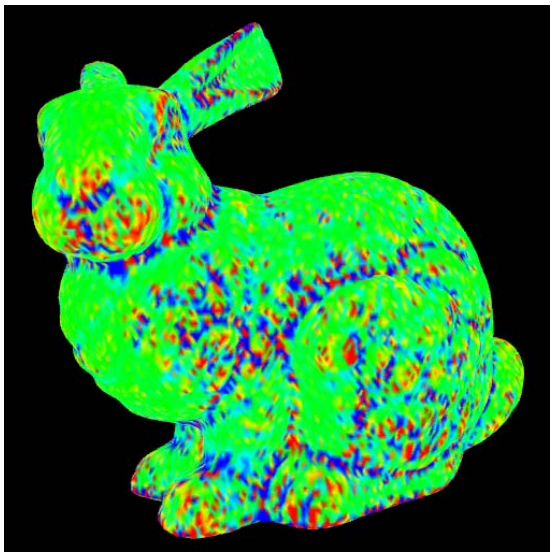


Figure 3: The Gaussian curvature approximation

3.5 Bonus (10%)

Implement “tangential smoothing” which moves vertices only in the tangent plane of the vertex, thus focuses (only) on enhancing triangle shapes. For this, you should project the uniform Laplace average back to the tangent plane of the vertex. Notice that you need to compute (via OpenMesh) and store the original normal of the vertex, in order to keep the vertices always on the original tangent plane, even after several tangential smoothing iterations.