

# Digital 3D Geometry Processing
# Exercise 3 - Surface Quality

Handout date: 09.03.2015

Submission deadline: 18.03.2015, 23:00 h

## 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:

- Hand in **only** the files you changed (headers and source). It is up to you to make sure that all files that you have changed are in the zip.
- A `readme.txt` file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Other files that are required by your `readme.txt` file. For example, if you mention some screenshot images in `readme.txt`, these images need to be submitted too.
- Submit your solutions to Moodle before the submission deadline.

## Goal

In this exercise you will implement the following tasks:

- Computing the uniform Laplacian approximation of the mean curvature;
- Computing the Laplace-Beltrami approximation of the mean curvature;
- Evaluating the triangle shapes of the input mesh;
- Approximating the Gaussian curvature.

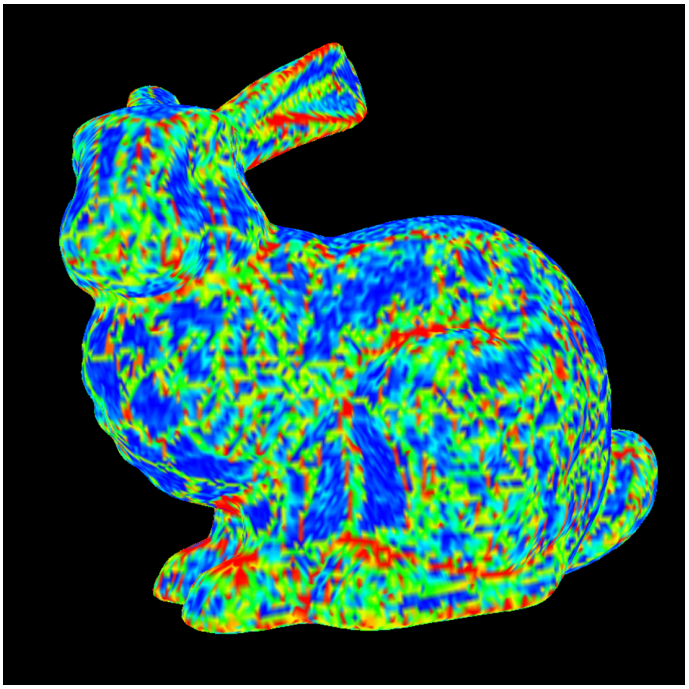


Figure 1: The uniform Laplacian approximation of the mean curvature at each vertex.

## Input

This exercise requires a triangular mesh as an input.

- `argv[1]` - a path to the input mesh `"bunny.off"`.

## Uniform Laplace curvature

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

$$L_U(v) = \frac{1}{N} \sum_{i}^{|N|} (v_i - v)$$

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

Implement the uniform Laplace approximation of the mean curvature in the function `calc_uniform_mean_curvature()`. Store the computed values in vertex properties `vunicurvature_`. Test your solution by loading the `bunny.off` model. To display the per-vertex mean curvature values right-click on the `"Surface Quality"` window and choose `"Uniform Mean Curvature"` option from the context menue. You should get the result similar to Figure 1.

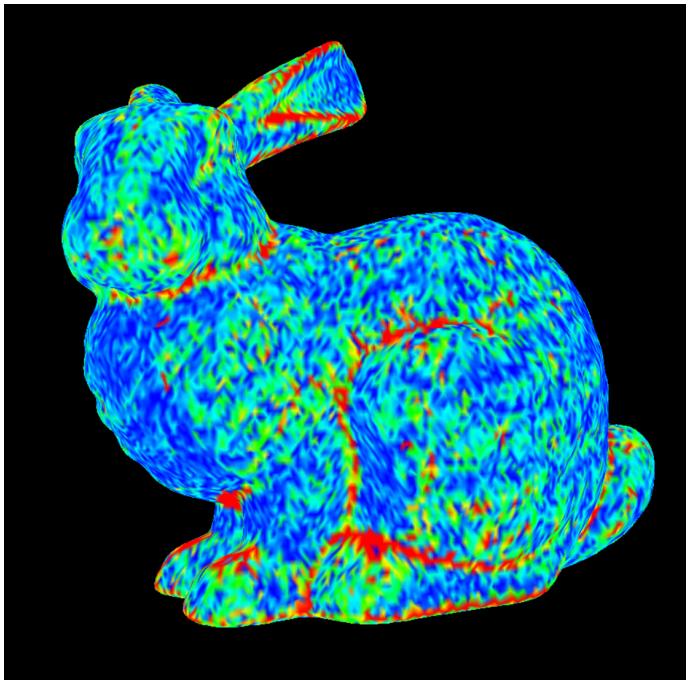


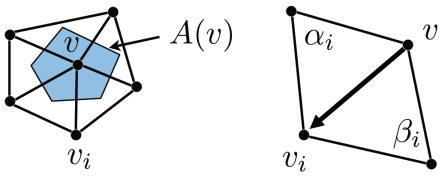
Figure 2: The Laplace-Beltrami approximation of the mean curvature at each vertex.

## Laplace-Beltrami curvature

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:

$$L_B(v) = \frac{1}{2A}\sum_{i}^{|N|}(\cot\alpha_i + \cot\beta_i)(v_i - v)$$

$$L_B(v) = w\sum_{i}^{|N|}w_i(v_i - v)$$




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.

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. To display

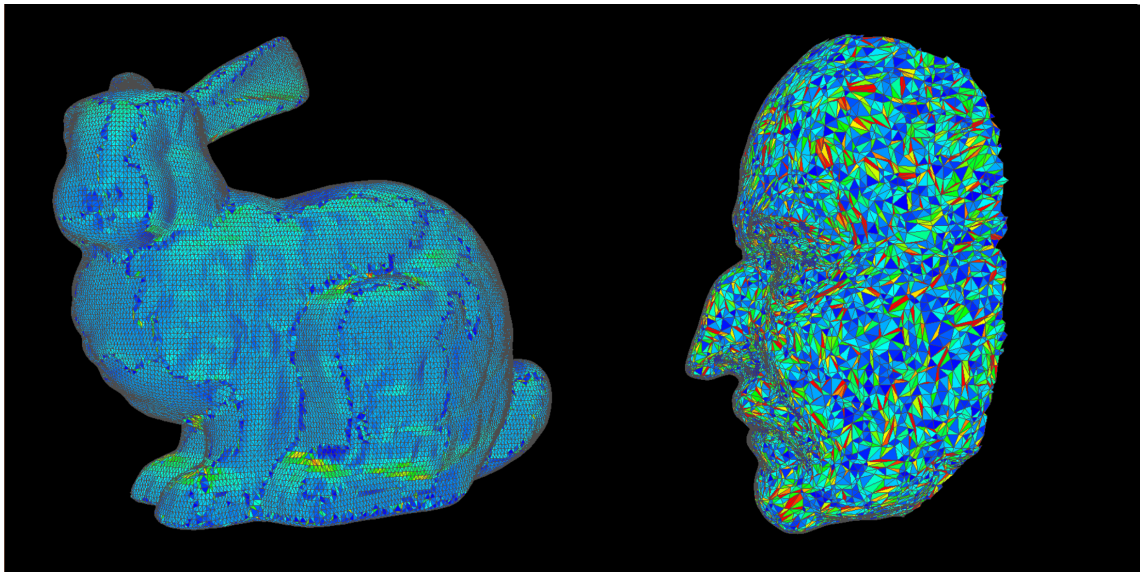


Figure 3: Triangles quality metric at each face.

the per-vertex mean curvature values right-click on the ”Surface Quality” window and choose ”Mean Curvature” option from the context menue. You should get the result similar to Figure 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 circumradius $r$, 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.

To display the per-face mean triangle shape metric right-click on the ”Surface Quality” window and choose ”Triangle Shape” option from the context menue. You should get the result similar to Figure 3.

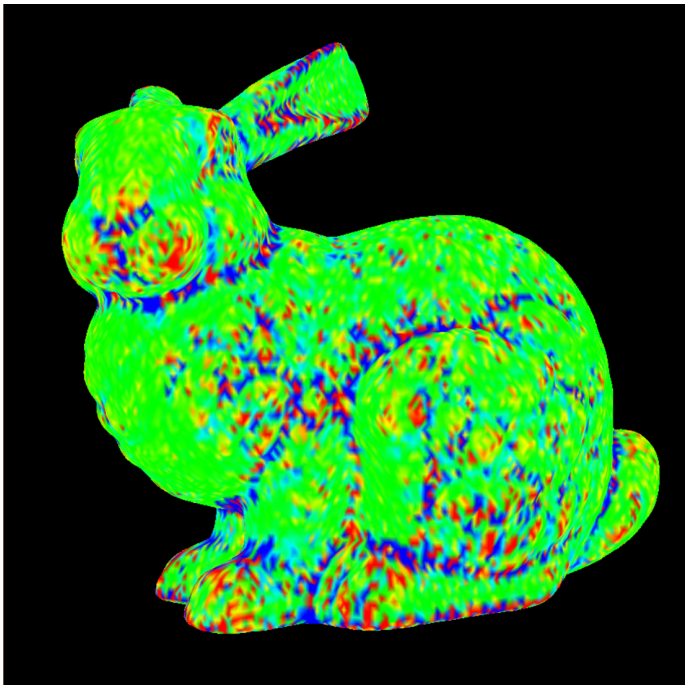
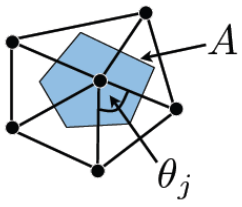


Figure 4: Approximation of the Gaussian curvature at each vertex.

## 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:

$$G = (2\pi - \sum_j \theta_j) / A$$




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 4.