UNIVERSITÄT BERN

Digital 3D Geometry Processing
Exercise 10 - Parameterization I

Handout date: 07.05.2019

Submission deadline: 14.05.2019, 13:00 h

Note

A .zip compressed file renamed to `Exercise`n`-GroupMemberNames.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 ILIAS before the submission deadline.

Goal

In this exercise you will implement harmonic functions on meshes by solving Lapcale equations. The next step is to find two harmonic functions whose contours form a quad-mesh-like network. You will generate edges that correspond to the isolines of harmonic functions using "marching triangles" method.

Solving Laplace Equations (4 pts)

Harmonic function maps each vertex to a scalar value. To compute these values solve a linear system $Ax = b$, where A is Discrete Laplace-Beltrami matrix, x contains the unknown values of harmonic function at the vertices. b is equal to 0 for all vertices except for the ones for which the values of harmonic function x_i are constrained.

In order to obtain a harmonic function that is not equal to a constant across the entire mesh, one needs to constrain the values of at least two vertices. Fix the values of harmonic function at the vertices i and j such that $x_i = 0$ and $x_j = 1$. To do this, replace the i-th and j-th row of A by $[0, ...0, 1, 0...0]$ with 1 at correspondingly at position i and j. Set b_i to 0 and b_j to 1.

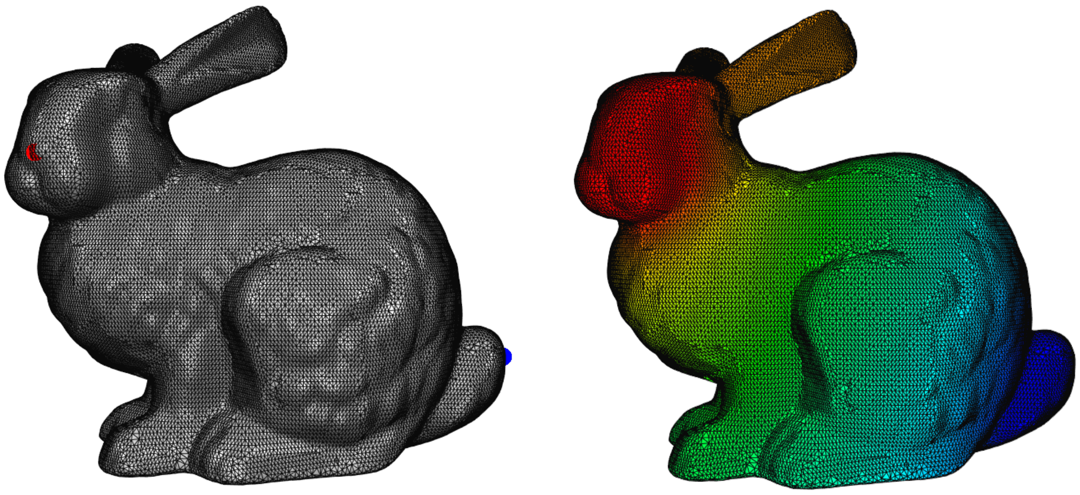

Figure 1: User interface for adding constraints.

Implement the described linear solve in the function `solve_harmonic_function()`.

Selecting Constrained Vertices

The framework provided with Exercise 9 offers an interface for selecting a vertex on the mesh. You do not need to implement anything for this part. Your task is to experiment with constraining different pairs of vertices and examine the resulting harmonic functions. You need to try two pairs of vertices one after the other, such that the level sets of the the two resulting harmonic functions form a quad-mesh-like network.

To add the pair of constraints

- Press the button `Show Constraint Vertices` to enter `Picking Mode` and show the default vertices.
- Select `Vertex 0` in the combo box, pick a vertex on the triangle mesh by left clicking on a vertex. The selected vertex will be shown as a red sphere.
- Select `Vertex 1` in the combo box, pick the other vertex. The second vertex will be shown as a blue sphere.
- Press the button `Solve Harmonic Function` to compute a harmonic function with the selected pair of vertices used as constraints.
- The function value is colored per vertex (see Figure 1).

Generating Edges at Isolines of Harmonic Functions (4 pts)

To compute the edges that correspond to the isolines of a harmonic function, first select the number of intervals, on the borders of which we are going to draw the isolines. Type the number of intervals in the corresponding window of the user interface. For example,

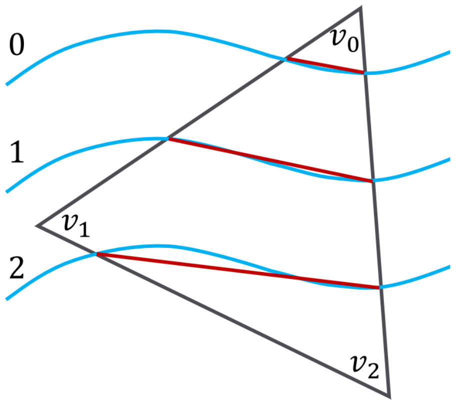

Figure 2: Generating Edges at Isolines of Harmonic Functions.

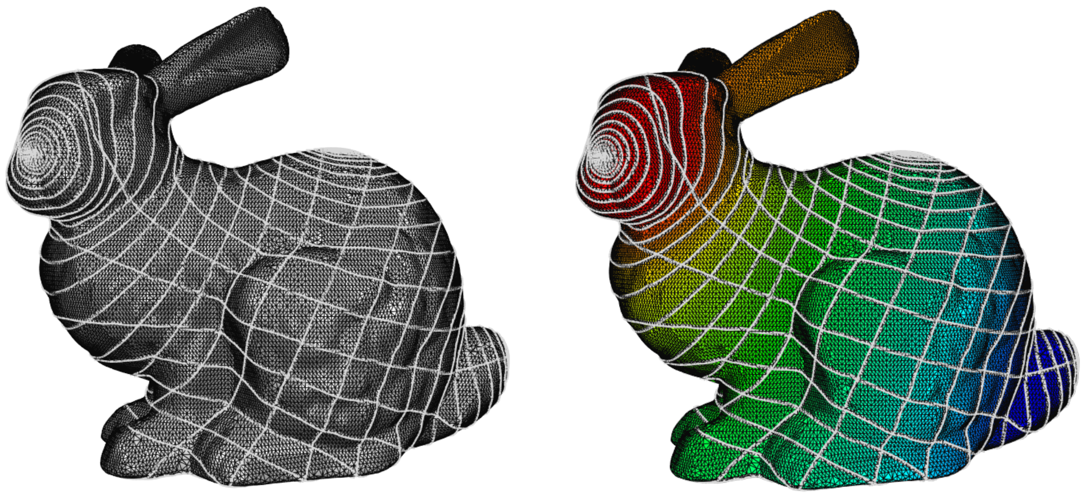

Figure 3: Isolines on Bunny mesh.

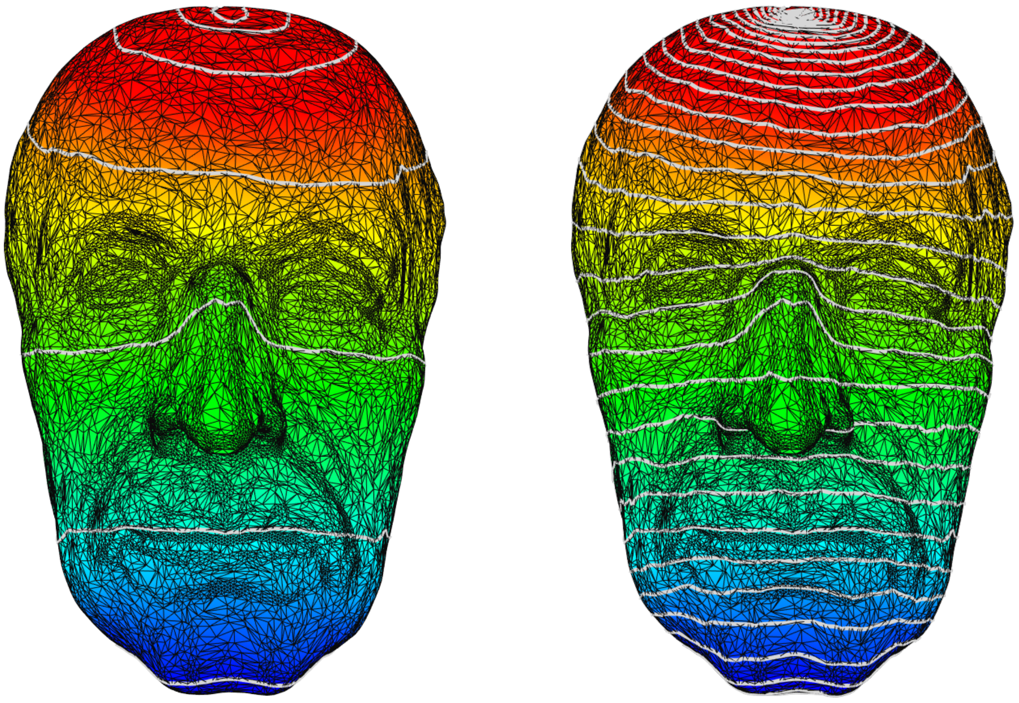

Figure 4: Varying the number of intervals.

if you choose 10 intervals, the values at interval borders will be 0.1, 0.2, ... 0.9, since the harmonic function varies between 0 and 1.

Given the values of harmonic function that correspond to the two vertices in triangle, find the first and the last interval border that fall between the isovalues at the two vertices. Use `std::pair` to return the indices of the first and the last interval border. For examples, for vertices v_0 and v_1 at Figure 2, it should return $(0,1)$, for vertices v_0 and v_2 - the pair $(0,2)$ and for vertices v_1 and v_2 it should return $(2,2)$. Implement this in the function `get_intervals_borders()`.

For each two edges of a triangle check if they are intersected by the same isoline. If this is the case, compute the intersections using linear interpolation of the isovalues. Add the resulting intersection points to the vector of `segment_points_` (this vector is used to create isolines afterwards). Implement this in the function `add_isoline_segment()`. To generate edges at isolines, press the button `Compute Isolines`.