CS468 - Geometry Processing Algorithms

Exercise 4 - Mesh Parameterization

Handout date: 11/8/2010

Submission deadline: 11/22/2010, 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 course.

What to submit

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 any problems encountered.
- Send your solutions to mirela@stanford.edu before the submission deadline.
  No late submissions will be accepted.

Reference Software

See 04-Parameterization.exe for an example of how your program should run.

Mesh Parameterization

In this exercise you will implement surface parameterization to be used for texture mapping. The tasks you will implement are as follows:

- Implementation of a simple disk parameterization method using uniform and discrete harmonic (i.e cotangent) barycentric coordinates with circular boundary.

- Use of parameterization for texture mapping.
- Computation of parameterization distortion as the overall distortion of triangle areas and angles.

Framework

A new project, Parameterization has been added to the framework from the previous exercise. It reuses files from the the ValenceViewer project. It adds an additional class HarmonicMapViewer which extends the MeshViewer to perform the parameterization computation as well as textured surface rendering. The project compiles to a command line tool that reads a mesh and provides visualization modes for a 3D mesh with and without texture mapping as well as flat mesh rendering for input texture visualization. The program accepts two parameters: the input mesh (.off) and the number of repeats, i.e. the number of times the texture map has to repeat itself. You will need to implement portions of the new class Parameterization.

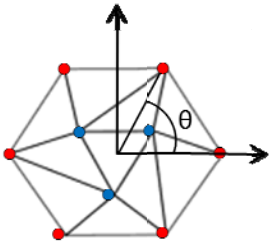
To solve the linear system you will need to use the gmm library, which is compiled with the project. Use:

- gmmVector B(n) to create a vector of size n.
- gmmMatrix A(n,n) to create a matrix of size nxn.
- A(i,j) to access the (i,j) location in the matrix.
- solve_linear_system(A, B, X) to find the solution to AX = B (defined in HarmonicMapViewer)

5.1 Uniform map

The uniform map defines the parameter of each vertex as the simple barycenter of the parameters of the vertices of its one-ring neighborhood. Let **x** and **y** be the vectors containing the parameter coordinates corresponding to all vertices of the mesh, B the set of boundary vertices associated to the disk boundary parameters $(\cos(\theta_i), \sin(\theta_i))$, and E the set of mesh edges. The coordinates of **x** and **y** are solutions of the systems:

$W\mathbf{x} = \mathbf{b}_x$ and $W\mathbf{y} = \mathbf{b}_y$, where $W_{ij} = \begin{cases} 1 & (i,j) \in E \\ -\sum_{i \neq j} w_{ij} & (i,i)\ i \notin B \\ 1 & (i,i)\ i \in B \\ 0 & otherwise \end{cases}$,



$b_{x_i} = \begin{cases} cos\ (\theta_i) & i \in B \\ 0 & otherwise \end{cases}$ and $b_{y_i} = \begin{cases} sin\ (\theta_i) & i \in B \\ 0 & otherwise \end{cases}$

Implement the calc_uniform_parameterization() function in the HarmonicMapViewer class. This function should fill the vparam_u_ vertex property with the corresponding vertex parameter using uniform map and fixed boundary.

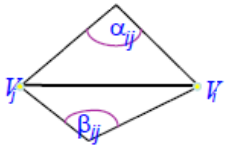
Write a routine to identify the boundary of the input mesh. Use arc length to place these vertices on the parameter disk boundary, i.e. the boundary arc length (= angle) in 2D should be proportional to the corresponding edge length on the 3D boundary.

Test your solution by loading the provided meshes. Press the U key to run the uniform parameterization and choose “UV Domain” to display the parameter disk.

5.2 Harmonic map

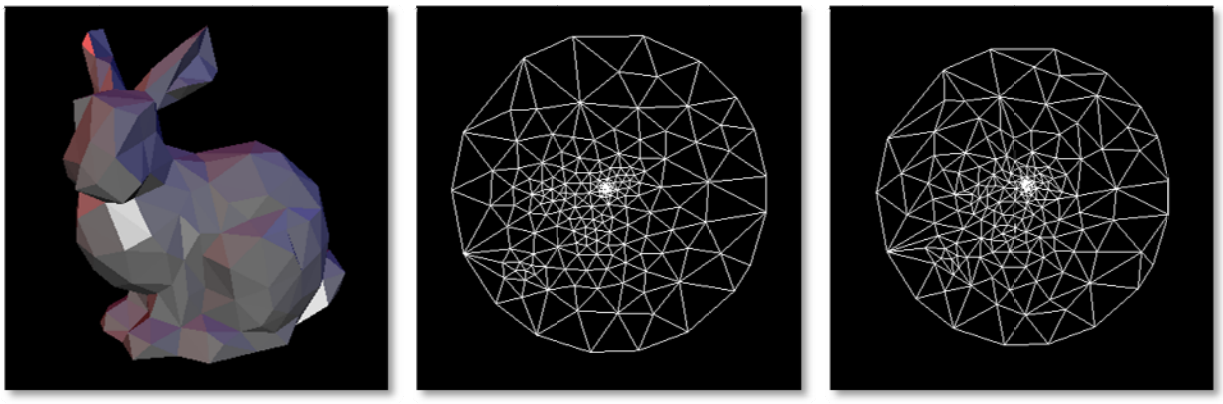
Uniform parameterization is easy to compute but does not contain any “shape information”. A harmonic map can be used instead to get better angle preservation by using the discrete harmonic weights W_{ij} defined as follows:

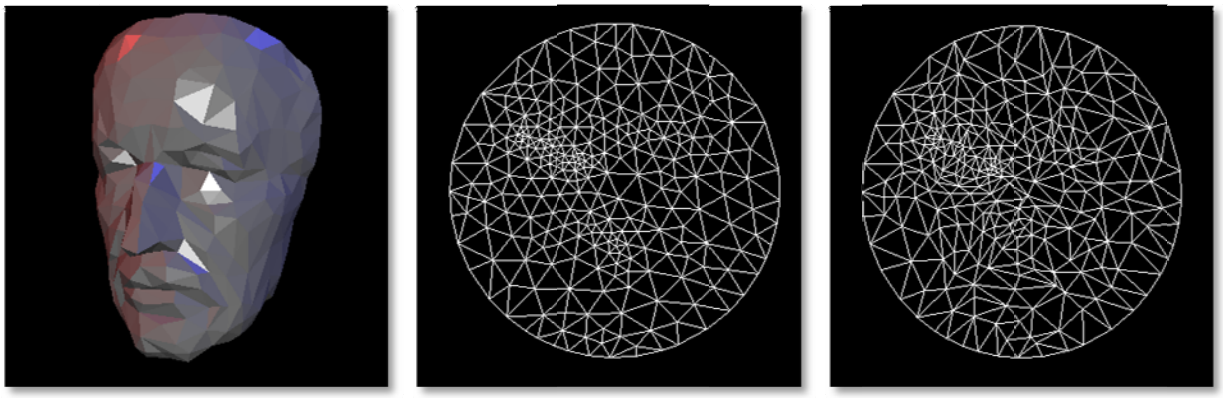
$$W_{ij} = \begin{cases} \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{2} & (i,j) \in E \\ -\sum_{i \neq j} w_{ij} & (i,i)\ i \notin B \\ 1 & (i,i)\ i \in B \\ 0 & otherwise \end{cases}$$



Implement the calc_harmonic_parameterization() function in the HarmonicMapViewer. This function should fill the vparam_h_ vertex property with the corresponding vertex parameter using a harmonic map and fixed boundary.

Test your solution on the provided meshes. Press the H key to run the harmonic parameterization.





(a) **(b)** **(c)**

Figure 1: (a) Input mesh, (b) uniform map, (c) harmonic map

5.3 Texture mapping

Complete the ComputeTextureCoordinates(int iTextureWidth, int iTextureHeight, int iRepeats) function which computes texture coordinates for a texture image of size iTextureWidth × iTextureHeight. This function should fill the texcoord_u_ and texcoord_h_ vertex properties with the texture coordinates corresponding to uniform map and harmonic map respectively.

Note that a standard texture has a rectangular 2D shape, hence the bounding box of the 2D image should be used as the (u,v) image of the mesh (so some of the texture will be wasted). The texture map may repeat itself periodically depending on the value of the iRepeats arguments.

Hint: for n repeats, the texture coordinate range should be [0..n].

Choose "Textured mesh" to display the input mesh with a checkerboard texture and the specified number of repeats. For the textured bunny with 10 repeats, you should get images similar to Figure 2.

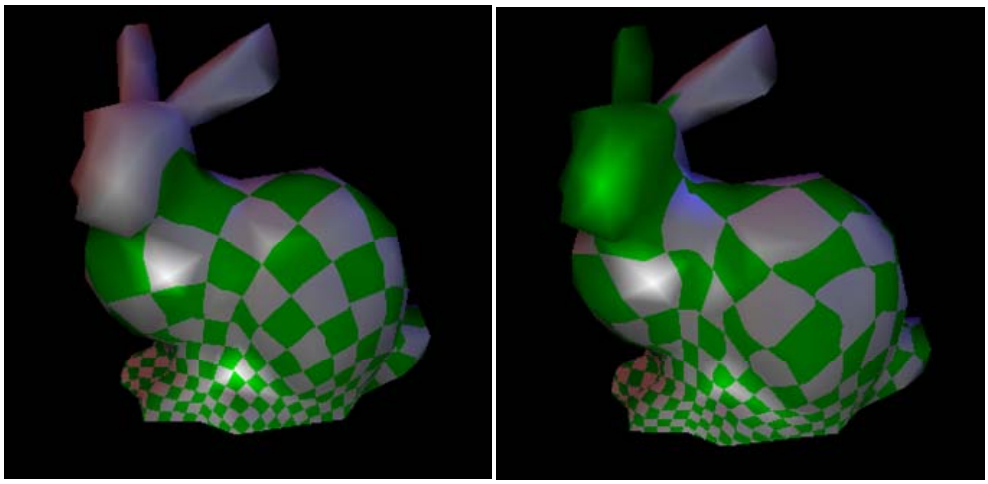


Figure 2: Textured bunny using harmonic map (left) and uniform map (right)

5.4 Parameterization distortion

The parameterization distortion can be measured using the overall (sum) distortion of triangle areas and angles.

Implement the calc_distortion() function which prints the angle and the area distortions for both parameterizations in the output window.

- For angles use the following formula $D_1 = \Sigma(\alpha - \beta)^2$ where α and β are the corresponding mesh angles in 3D and 2D respectively (the number of angles is triple the number of faces).
- For areas use the following formula $D_2 = \Sigma(a - b)^2$ where a and b are the normalized areas of corresponding triangles in 3D and 2D. Each of the areas should be normalized by the total area of the 3D or 2D domain, respectively.

Compare the uniform parameterization distortion and the harmonic parameterization distortion on the provided inputs.