Assignment 4: Mesh Parameterization

Handout date: 12.04.2024
Submission deadline: 03.05.2024 at 10:00

In this exercise you will

- Parameterize a mesh by minimizing four different distortion measures,
- with fixed or free boundaries.
- Visualize the distortion by color coding.

The majority of this task involves setting up a sparse linear system and solving it to obtain the uv coordinates of the parameterization. The specific linear system will depend on the type of parameterization and desired boundary conditions. To this end, you are provided with a function that computes the gradient matrices, but you will have to derive the systems, based on the distortion measure, on your own. The parameterization energies to be implemented in this assignment are:

- Spring energy (uniform Laplacian)
- Dirichlet/harmonic energy (cotangent Laplacian)
- Least Squares Conformal Maps (LSCM)
- As-Rigid-As-Possible (ARAP)

1. Setting up the boundary conditions **[4 points]**

The first task is to define the boundary conditions. There are a few possible options:

- The boundary of the mesh is fixed to a unit disc.
- The boundary is free, but two vertices are fixed based on a strategy.
- The boundary is free, but a few constraints are added to only fix the degrees of freedom of the parameterization.

1.1. **Finding the fixed vertex indices and their positions.** In both cases you are required to find the lists of indices of the fixed vertices and their fixed positions in the plane. For the first option, you can use the libigl functions `boundary_loop` and `map_boundary_to_circle`. For the second option the boundary should not be fixed, however, in order to get a unique solution, two vertices must be fixed. Note that this is only relevant for the LSCM and ARAP parameterization, as the spring and Dirichlet energies can only be used in the fixed boundary setting. Try to explore various possibilities for picking these two vertices and devise a strategy that will result in nice parameterization (hint: it is a good practice to pick the two most distance vertices as the fixed ones).

Relevant `libigl` *functions:* `boundary_loop`, `map_boundary_to_circle` as mentioned and perhaps `dijkstra` .

1.2. **Fixing only the necessary degrees of freedom.** In the free boundary case (for LSCM and ARAP), fixing two vertices results in reducing more degrees of freedom than necessary to make the system full rank. This can lead to non-optimal results when the choice of fixed vertices is not ideal, as the system becomes unnecessarily over-constrained. Find a way to alleviate this issue by reducing the number of constraints so that they only eliminate the necessary degrees of freedom of the parameterization.

1.3. **Convert the boundary conditions to linear constraints.** in order to satisfy the boundary conditions, they have to be described as a linear system in the following way,

$$C \begin{pmatrix} u \\ v \end{pmatrix} = d$$

Implement the function `ConvertConstraintsToMatrixForm` which converts the lists of fixed vertices and their positions to a sparse matrix C and a vector d.

2. Write the parameterization problem in matrix form and construct the matrix **[10 points]**

As was shown in the tutorial, all parameterization methods discussed in this course solve a system of the form

$$\begin{pmatrix} A & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ d \end{pmatrix}. \tag{1}$$

The matrix C and vector d were found in the previous part. In this part you will find the matrix A and vector b relevant for the specific parameterization method. Initiating each type of parameterization can be done by pressing '1'-'4'.

2.1. **Uniform and cotangent Laplacian. [2 points]**

In this case A is constructed using the matrix L where L can be either the uniform or cotangent laplacian, and $b = 0$. Pay attention to the dimensions of the matrices and vector! it will help you to debug your code. You can use the function `igl::adjencency_matrix` and `igl::cotmatrix` to help you.

2.2. **LSCM. [3 points]**

One of the ways to define the LSCM distortion measure is as follows,

$$D(J) = \|J + J^T - (trJ)I\|_F^2$$

Follow the technique you saw in the tutorial to derive the system required to minimize the LSCM distortion. You are provided with the function `computeSurfaceGradientMatrix` which computes the gradient matrices D_x, D_y as shown in class. Don't forget to include the triangle areas!

Sanity check!: it was shown in [2] that the minima of the dirichlet and LSCM energies are the same when the boundary is fixed.

Relevant `libigl` *functions:* `double_area`, `cat`.

2.3. **ARAP. [5 points]**

The ARAP distortion is defined by

$$D(J) = \|J - R\|_F^2 \tag{2}$$

where R is the closest rotation matrix to J. Since R is non linearly dependant on J, this distortion is not quadratic, and hence cannot be minimized by solving a single linear system. The *local/global* approach proposed in [1] is an iterative approach for minimizing the ARAP distortion. Starting from an initial guess (for example, obtained via LSCM) the idea is to iterate the two following steps:

- Local step: The Jacobians for each face of the current iterate are computed. Then for each Jacobian the closest rotation matrix is found. This can be done using the SVD of J as shown in the lecture.
- Global step: Once the closest rotation for each Jacobian is found, they are all assumed to be fixed, and then (2) can be minimized by solving a linear system.

Use the provided function `SSVD2x2` to find closest rotations as mentioned above, and derive the matrix form of (2). Then solve the linear system (next part) to obtain a parameterization with lower ARAP distortion. Keep pressing '4' to initiate the parameterization and get better result. Stop when there is no observable improvement. For more details refer to [1].

3. Construct the system and display the results **[2 points]**

3.1. **Solve and show the parameterization.** Once A, C, b and d are found, construct the system in (1) and solve it using the Eigen solver `Eigen::SparseLU`. Take the relevant part of the solution (i.e. without the Lagrange multipliers) and store it in global variable `UV`. You can now see the parameterization on the left side of the screen, and a checkerboard texture on the mesh (lifted via the parameterization). Use '+' and '-' to scale the texture to appropriate size (or use the GUI option to adjust the scale).

3.2. **Visualize the distortion.** Color code the distortion of the faces to visualize the quality of the results. Experiment with different criterions (angle preservation, edge length preservation, etc.). Highly distorted triangles should appear in red and undistorted triangles in white.

Live demo. During the live demo you are expected to be able to demonstrate:

- The results of the four parameterizations with fixed and free boundaries.
- Each of the distortion measures for each of the parameterizations.

References

[1] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A local/global approach to mesh parameterization. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1495–1504, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[2] Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. Spectral conformal parameterization. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1487–1494, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.