Assignment 6: Final Project

Handout date: 05.05.2017
Submission deadline: 02.06.2017 at 08:00
Demo session: 02.06.2017 08:00-10:00

In this exercise you will implement one of the 7 algorithms explained in the following sections. No additional code and/or precise instructions will be provided for this exercise, you will have to read the corresponding paper, understand and implement it. You can obviously use `libigl` and you can reuse any of the code from the previous exercises.

In the following we describe 7 possible projects, but we are open to suggestions if you want to implement something else that you like. Contact the TAs by email describing your project BEFORE starting to work on it.

The projects are the following:

(1) (Deformation) As- Rigid-As-Possible Shape deformation [7].
(2) (Deformation) Cage-based deformation with Mean Value Coordinate [4].
(3) (Remeshing) Interactive Geometry Remeshing [1]
(4) (Discr. Diff. Geom.) Geodesics in Heat [2]
(5) (Parameterization) Bijective Parameterization with Free Boundaries [6]
(6) (Diff.Quantities.) Smooth feature lines on surface meshes [3]
(7) (Interpolation) Multi-scale biharmonic kernels interpolation [5]

For all projects, you have to write a detailed report with all images of the results and a description of what you implemented and commit everything to the assignment 6 folder in your repositoryo.

Project 1 - (Deformation) As- Rigid-As-Possible Shape deformation

Short Description. You will implement the complete ARAP deformation algorithm described in [7]. The method allows to deform triangular meshes by moving handles that are groups of mesh vertices.

Implementation details. You can use the UI of Exercise 5 with only a small addition: it must be possible to select the number of ARAP iterations performed at every step in the UI. ARAP is slower then the simple linear deformation algorithm implemented in Exercise 5, but your software must be able to deform interactively meshes with less than 10k vertices.

Additional libraries required. No external libraries are required.

References. [7]

Project 2 - (Deformation) Cage-based deformation with Mean Value Coordinates

Short Description. You will implement the computation of mean value coordinates for arbitrary 2D polygons [4], and use these coordinates to deform images in real-time.

Implementation details. You need to be able to visualize an image and edit it interactively. You can use the igl viewer to render the image with opengl by plotting a square of the size of the image (as a pair of two triangles) with proper UV coordinates associated with its vertices.

Follow the pseudocode in [4] to implement the 2D mean value coordinates and implement a simple UI that allows the user to draw a polygon and change the position of its vertices interactively. Every change of the polygon will also deform in real time the corresponding image. This is achieved by following the algorithm explained in Section 6.2 of [4].

Additional libraries required. No external libraries are required.

References. [4]

Project 3 - (Remeshing) Interactive Geometry Remeshing

Short Description. You will implement a simplified version of the remeshing algorithm presented in [1]. The algorithm will be able to re-mesh any surface with a single boundary.

Implementation details. Follow the original paper for the implementation, with the following simplifications:

- Implement it only for the case of a simple open mesh with a single boundary, parametrize the entire mesh in a single patch.
- If you want, use the harmonic or LSCM parametrization that you implemented in Exercise 4 instead of the conformal parametrization used in the paper. Note that you have to map the boundary of the surface to a square in the UV plane instead of a circle.
- Implement only the area distortion and curvature maps.
- Use Floyd-Steinberg dithering algorithm to generate the samples (http://en.wikipedia.org/wiki/Floyd-Steinberg_dithering).

Add two widgets in the UI that allow to change the number of samples and the importance maps to use (area distortion or area distortion + curvature).

Your software needs to be able to remesh surfaces uniformly (area distortion) or adaptively, combining the area distortion map with the curvature map.

Additional libraries required. For generating the delaunay triangulation on the plane use the library "triangle" (http://www.cs.cmu.edu/~quake/triangle.html)

References. [1]

Project 4 - (Discr. Diff. Geom) Geodesics in Heat

Short Description. Implement the algorithm to compute geodesic distance presented in [2]. Extend the UI to be able to pick a point on the surface and plot the distance between every other point and the selected one.

Implementation details. The UI must be extended to be able to pick a single point. Try to precompute everything that you can to make the computation as fast as possible. When a point is selected, compute the geodesic distance between every point of the surface and the selected one. Plot the distance as a scalar field over the surface, and also plot isolines. Add to the UI the parameter used by the method, experiment with it and discuss the effect.

Additional libraries required. No external libraries are required.

References. [2]

Project 5 - (Parameterization) Bijective Parameterization with Free Boundaries

Short Description. You will implement the single patch parametrization algorithm described in [6].

Implementation details. Use can use the framework developed for assignment 4. Follow the original paper, which consists of three parts:

- Initialization: Use the uniform parameterization algorithm that you implemented in that assignment.
- Descent direction: The algorithm uses the L-BFGS method (see http://en.wikipedia.org/wiki/Limited-memory_BFGS), which uses the gradient to find a descent direction (similar to gradient descent, but usually better). You are required to compute the gradient and implement the L-BFGS method.
- Line search: The algorithm takes the descent direction and find a step size that decreases the distortion and is guaranteed not to introduce flipped triangles.

Important! You are not required to implement the intersection free parameterization in section 3.2 of the paper.

Additional libraries required. No external libraries are required.

References. [6]

Project 6 - (Diff.Quantities.) Smooth feature lines on surface meshes

Short Description. Extraction of feature lines from triangulated surfaces.

Implementation details. Follow the paper until section 4.2, do not implement section 5.

For the visualization of the lines, you can use the rendering facilities included in the provided viewer.

Additional libraries required. No external libraries are required.

References. [3]

Project 7 - (Interpolation) Multi-scale biharmonic kernels interpolation

Short Description. Scalar field interpolation on surfaces is a useful geometry processing tool. You will implement a recent interpolation algorithm that relies on the computation of bounded biharmonic functions.

Implementation details. [5] describes two types of biharmonic kernels, called biharmonic and pre-biharmonic. You should implement only the biharmonic kernels. The kernels will then be used as radial basis functions to interpolate scalar values specified in a sparse set of points on the mesh, following the algorithm described at the beginning of Section 7 in [5].

You have to implement a simple UI that allows to select points on a mesh and to associate a scalar value to them. The system should compute and show in real-time the interpolant scalar field over the entire surface using pseudo-colors.

Additional libraries required. You need a Quadratic Program Solver to compute the biharmonic kernels, we strongly suggest to use Mosek. You can get a free academic license using your ETH email account from http://www.mosek.com/.

References. [5]

Optional exercise (Bonus: 10%)

Implement an additional project from the previous list.

References

[1] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. *ACM Trans. Graph.*, 21(3):347–354, July 2002.

[2] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph.*, 2013.

[3] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. Smooth feature lines on surface meshes. In *Proceedings of the third Eurographics symposium on Geometry processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[4] Kai Hormann and Michael S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.*, 25(4):1424–1441, October 2006.

[5] Raif M. Rustamov. Multiscale biharmonic kernels. *Computer Graphics Forum*, 30(5):1521–1531, 2011.

[6] Jason Smith and Scott Schaefer. Bijective parameterization with free boundaries. *ACM Trans. Graph.*, 34(4):70:1–70:9, July 2015.

[7] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, SGP '07, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.