Transfinite Surface Library

ShapEx

July 14, 2015

1 Conventional Geometry

Conventional geometric entities are provided by a shared library, and are accessed through the header file `geometry.hh`. The main classes defined there are as follows:

- `Vector2D`, `Vector3D`: 2- and 3-dimensional vectors, with the usual operators.
- `Point2D`, `Point3D`: Aliases for `Vector2D` and `Vector3D`.
- `DoubleVector`, `Vector2DVector`, `Point2DVector`, etc.: Aliases for STL vectors containing the respective objects.
- `BSCurve`: A (non-rational) B-spline curve.
- `BSSurface`: A (non-rational) B-spline surface, with or without trimming. This is a dummy class only for data transfer.
- `TriMesh`: A triangle mesh, capable of writing Wavefront `.obj` files via `writeOBJ`(*filename*).
- `IGES`: An IGES file writer, exporting the function `writeSurface`(*surface*).

2 Transfinite Surfaces

The general interface for handling transfinite interpolation surfaces is in the header file `surface.hh`. It defines the abstract class `Transfinite::Surface`, which provides several functions (detailed below). For actual use, one has to use a child class—presently there are four options: `SurfaceSideBased`, `SurfaceCornerBased`, `SurfaceGeneralizedCoons`, `SurfaceCompositeRibbon`, corresponding to the patch types SB, CB, GC and CR in [1], respectively. One- and two-sided surfaces are not supported.

2.1 Surface Details

Transfinite interpolation patches have several constituents, such as ribbon surfaces, domain polygons, parameterization mappings, or blending functions. The above surface types already define the blending functions, but there are several choices in all of the other parameters.

In this library, we have only one ribbon type, implemented in `RibbonCompatible`, which can be formulated as follows (using the notations of [1]):

$$R_i(s,d) = P_i(s) + \left[-P'_{i-1}(1)^{\perp} \cdot (1-s) + P'_{i+1}(0)^{\perp} \cdot s\right] \cdot d,$$

where $v^{\perp}$ is the vector v projected into the tangent plane at the current point, i.e.,

$$v^{\perp} = v - n(vn),$$

n being the unit normal vector obtained by a rotation minimizing frame [3] between the normals of the end vertices.

There are two domain polygon computations implemented in the library. The simplest is `DomainRegular`, which uses a regular n-sided polygon as the domain. Currently all surface types employ this method. The other one is `DomainCircular`, where an n-sided cyclic polygon is constructed in such a way, that the central angles and the arc lengths of the boundary curves have a constant ratio [2].

The SB and CB patches use bilinear parameterization (`ParameterizationBilinear`), and GC uses bilinear-based interconnected parameterization (`ParameterizationInterconnected`). For CR patches, the parameterization is based on Wachspress coordinates λ_i ($i \in [1 \dots n]$):

$$\begin{aligned} s_i &= \lambda_i / \left(\lambda_{i-1} + \lambda_i\right), \\ d_i &= 1 - \lambda_{i-1} - \lambda_i, \end{aligned}$$

implemented in `ParameterizationBarycentric`. This computation of the s parameter is not published yet. Otherwise, for details on these mappings, see [1].

2.2 Surface Creation

After creating a surface object of the desired type, the following steps finalize the surface:

1. Set the boundary curves using `setCurves`(*curves*). Alternatively curves can be set one by one, using `setCurve`(i, *curve*) to set the i-th curve.

2. Call `setupLoop`() to normalize the curves and fill in adjacency information. This function has to be called every time a new curve is assigned to the surface.

3. Call `update`() to generate ribbons and clear the cache. This function has to be called every time the curves have been changed. If only the i-th curve has changed, it is sufficient to call `update`(i).

4. Use of the γ function [1] can be turned off by `setGamma(false)`. It is on by default.

2.3 Evaluation

A point p in the 2D domain can be evaluated by calling `eval`(p). For convenience, there is an overloaded function that evaluates the surface using a given resolution, thereby creating a triangle mesh: `eval`(*resolution*).

2.4 Fitting

There are two options for fitting quadrilateral B-spline surfaces.

Central Split

This method splits the n-sided transfintie surface into n quadrilaterals, and fits a B-spline surface on each of them. (Except in the 4-sided case, where only one B-spline surface is used.) The function for this is `fitCentralSplit`(*ftol*, *ktol*, *density*). The parameters are:

- *ftol* is the main fitting tolerance to be achieved.
- *ktol* is the knot snapping tolerance, which is the smallest allowed difference between two knot values. As very close knots can cause artefacts, we recommend setting this value to at least 0.01 (curves are normalized to the interval [0,1]).
- *density* is the sampling density for the subdividing curves, the default is set to 30.

Trimming

This method fits a larger B-spline surface on the n-sided patch, and trims the sides with the original boundary curves. The function for this is `fitTrimmed`(*ftol*, *resolution*, *maxu*, *maxv*, *wcurv*, *wosc*). The parameters are:

- *ftol* is the main fitting tolerance to be achieved.
- *resolution* is the surface's sampling resolution, as in `eval`. Defaults to 15.
- *maxu* and *maxv* are the maximum number of control points in the u and v directions, respectively. These are needed because of the idiosyncrasies of the fitting algorithm. The default is 12 in both directions.
- *wcurv* and *wosc* are the curvature and oscilliation minimization weights, respectively. The default is 1e-6 for both.

References

[1] P. Salvi, T. Várady, A. Rockwood, *Ribbon-based Transfinite Surfaces*. Computer Aided Geometric Design, Vol. 31(9), pp. 613–630, 2014.

[2] T. Várady, A. Rockwood, P. Salvi, *Transfinite Surface Interpolation over Irregular n-sided Domains*. Computer Aided Design, Vol. 43(11), pp. 1330–1340, 2011.

[3] W. Wang, B. Jüttler, D. Zheng, Y. Liu, *Computation of Rotation Minimizing Frames*. Transactions on Graphics, Vol. 27(1), p. 2, 2008.