Planar Parameterization of Triangulated Surface Meshes
Proposal for a CGAL package

Pierre Alliez   Bruno Lévy   Laurent Saboret

Keywords: parameterization, bijective mapping, triangulated meshes, convex embedding, conformal, harmonic maps, free boundary methods.

1 Introduction

Parameterizing a surface amounts to finding a one-to-one mapping from a suitable domain to the surface. A good mapping is the one which minimizes either angle or area distortions in some sense. In this proposal we focus on triangulated surfaces that are homeomorphic to a disk and on piecewise linear mappings into a planar domain. Although the main motivation behind the first parameterization methods was the application to texture mapping, it is now frequently used for mapping more sophisticated modulation signals (such as normal, transparency, reflection or light modulation maps), fitting scattered data, re-parameterizing spline surfaces, repairing CAD models, approximating surfaces and remeshing.

2 Proposal

We propose to elaborate upon a CGAL Package which implements some of the state-of-the-art parameterization methods, such as LSCM (Lévy et al. 02), fixed or free boundary conformal maps, Authalic, Floater mean coordinate values or Tutte uniform weights. The package will propose an interface with both the 2D Triangulation Data Structure enriched with 3D points and the Polyhedron. Since parameterizing meshes require efficient representation of sparse matrices and efficient iterative or direct linear solvers, we plan to evaluate the standard packages or propose a separate package devoted to linear algebra.

3 Intended Audience

The intended audience of this package is researchers, developers or students developing algorithms around parameterization of triangle meshes for geometry processing as well as for signal mapping on triangulated surfaces.

4 Specifications

4.1 Input

- A triangulated mesh:
  - 2-manifold
  - Oriented
  - One connected component. The input mesh can be of any genus, -but- it has to come with a description of a boundary (a list of oriented edges given by a set or vertices) which is the boundary of a topological disc. If no boundary is given, we assume that it coincides with the unique boundary already in the input mesh. Note that this way the user is responsible for cutting a closed mesh of arbitrary genus (even a topological disc with an intricate seam cut), as long as this condition is verified.
- A set of constraints (a constraint specifies two u,v coordinates for each instance of a vertex along the boundary).
  - For free boundary methods: only two constraints (the pinned vertices). They have to be on the specified boundary.
  - For fixed boundary methods:
    - ∗ a list of constraints given by the user. The whole boundary has to be specified.
    - ∗ one convex shape specified by:
      - · one shape among a set of standard ones (circle, square, equilateral triangle).
      - · a convex polygon.
      - · for both cases above the user can also select a boundary parameterization among two common methods: uniform or arc-length parameterization.

4.2 Output

One uv coordinate for each interior vertex, and one uv coordinate for each instance of a vertex along the input boundary. It can be *e.g.* output in a list of 2D points through an output iterator.

4.3 Features

- Parameterizations:
  - Fixed boundary:
    - Tutte uniform weights (guaranteed one-to-one mapping for convex boundary).
    - Mean coordinate values (ditto).
    - Discrete conformal maps (conditionally guaranteed if all weights positive and convex boundary).
    - Authalic (ditto).
    - Others? (e.g. inverse, centripetal).
  - Free boundary:
    - Least squares conformal maps.
    - Discrete conformal maps.
- Constraint builder: for fixed boundary methods, generates a set of constraints.
- Mesh data structure: it has to be interfaced with the CGAL 2D Triangulation Data Structure (enriched with 3D points) and 3D Polyhedron.
- Preconditions (may be turned off):
  - assert triangle mesh (for polyhedron).
  - assert convex polygon for fixed boundary methods.
  - assert genus (does the specified boundary bound a topological disc?).
- Postconditions:
  - assess all positive coefficients (to ensure one-to-one mapping)
  - assess matrix conditioning (reliability of solution).
- Linear Algebra:

  Although they are often tightly coupled, there are 2 main components:
  - Data structure for sparse matrices
  - Linear solver (direct or iterative)

  The CGAL parameterization package will be loosely linked to the solver. Replacing it will be easy.

  Which linear solver?
  - OpenNL (Bruno Lévy) works very well but we may not be allowed to incorporate it in CGAL (GPL license and rights to check with VSP Technology). To be added to CGAL as a separate package?
  - GSL is GPL and is not optimized for sparse matrices => we cannot use it
  - dD kernel linear algebra is not optimized for sparse matrices => we cannot use it
  - BOOST::ublas: we cannot use it has the sparse matrix structure cannot grow dynamically (according to Bruno Lévy).
  - A quick search on Internet gave us dozens of sparse linear solvers that are free and with a license compatible with CGAL. The next ones seem promising (needs further investigation):
    - The NIST Sparse BLAS
    - SuperLU
    - TAUCS

References

[DMA02] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, September 2002.

[FH05] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer, Berlin, Heidelberg, 2005.

[Flo03] Michael Floater. Mean value coordinates. *Computer Aided Design*, 20(1):19–27, 2003.

[LPRM02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In *Proceedings of the 29th Conference on Computer Graphics and Interactive Techniques SIGGRAPH*, volume 21(3) of *ACM Transactions on Graphics*, pages 362–371, 2002.