Parameterization of Discrete Surfaces

Release 0.01

Arnaud Gelas[1], Alexandre Gouaillard[2]

November 21, 2007

[1]Biomedical Imaging Laboratory, SBIC, A-STAR, Singapore
[2]Center For Excellence in Genomic Sciences,
California Institute of Technology, Pasadena, USA

Abstract

Parameterization of surfaces, sometimes called "'flattening"' as it maps a surface embedded in 3D into its intrinsic 2D domain, is a powerfull tool for the analysis of surfaces. For the past years, there has been a growing interest in the Community that even lead to one implementation of one special type of parameterization in ITK [1,5,8,9]. We are providing here a more general framework for parameterization of single connected surfaces of any genus. It is based on a recent addition to ITK: itkQuadEdgeMesh [6] which allows an elegant an optimal implementation of algorithm for geometry and topology processing of discrete 2-manifolds. The 6 algorithms that we implemented map the meshes into a planar domain with fixed boundary leading to more stability and speed than mapping into the spherical domain. Each of them use different kind of parameterisation with different properties. The conformal parameterisation is usually used as it is intrisic to the geometry of the mesh and thus allow shape analysis independently of the connectivity. However if flatening the mesh is your only goal, then the simplest parameterization algorithm (graph theory) will give you the best speed. This is the case when you want to further process the mesh in a lower dimension. Using specific solvers, the parameterisation of meshes can be done sufficiently fast (couple of seconds) to consider this approach in interactive applications.

Contents

1 Introduction

1.1 Statement of the problem

Given any two meshes with the same topology, it is possible to compute a one-to-one and onto mapping between them [11]. Whenever one of these two surfaces is represented by a triangular mesh, with at least one boundary and singly connected, this problem is referred as *mesh parameterization*. Note that a surface without boundary can be cut open along an arbitrary path to lead to the same surface with a boundary. An algorithm to generate such a "'cut -graph"' is allready included in the itkQuadEdgeMesh [6] submission. Thus the problem we adress with this code is much more general than the one adressed in [1, 5, 8, 9] where they limit their application to genus 0 meshes, and where they map to sphere, which is known to be more unstable.

1.2 Applications

- detail mapping, synthesis
- morphing, detail transfert
- mesh completion
- mesh editing
- remeshing
- compression
- surface fitting
- etc...

This problem have received a lot of interest over the two last decades. Here we will not review all of them, so we invite interested reader to have a look to [4, 11, 14] for recent surveys concerning all these techniques. Parameterization, sometimes called "'flatening"', of surfaces has been used in brain mapping [7], Colonoscopy [10], cortical surface shape analysis [12], vessel geometry processing and many other medical or biological problems. In this paper, we focus on *linear parameterization with fixed boundaries*.

In the following sections, we first present the required theoretical background and present in details the design of our code.

Figure 1: A 3D 1-ring and its associated flattened version.

2 Background

2.1 Principle

Since we only consider *linear parameterization with fixed boundaries*, one can see this method as a *spring and mass model* where each vertex of the mesh is a mass and each edge of the triangular mesh is a spring. By constraining the boundary of this model, the interior vertices will relax to the minimum of spring energy.

Consider each spring is ideal, *i.e.* the rest length is null and the potential energy is only $\frac{1}{2}Dl^2$, where D is the spring constant and l is the length of the spring. Then specify the desired parameter values $\mathrm{u}_i = (u_i, v_i)_{i=1,\dots,L}$ for the boundary points $\mathrm{p}_i \in \mathcal{B}$, with L is the number of points on the boundary, then compute the parameterization can be computed by minimizing the following energy:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i}\frac{1}{2}D_{ij}\|\mathrm{u}_i - \mathrm{u}_j\|^2, \tag{1}$$

where N is the number of points of the mesh, $\mathcal{N}_i$ is the neighborhood of the vertex p_i, and $D_{ij} = D_{ji}$ is the spring constant corresponding to the edge connecting p_i and p_j.

The minimization of Eq. 1 leads to solve the following sparse linear systems

$$A \cdot \mathrm{U} = \bar{\mathrm{U}} \tag{2}$$

where A is a $n \times n$ sparse square matrix with elements

$$A_{ij} = \begin{cases} 1 & \text{if } i = j, \\ -D_{ij} \Big/ \displaystyle\sum_{k\in\mathcal{N}_i} D_{ik} & \text{if } j \in \mathcal{N}_i, \\ 0 & \text{else} \end{cases} \tag{3}$$

$\mathrm{U} = (\mathrm{u}_i)_{i=1,\dots,N} = (u_i, v_i)_{i=1,\dots,N}$ is a $2 \times n$ matrix of unknown parameter values (inner points) and $\bar{\mathrm{U}} = (\bar{\mathrm{u}}_i)_{i=1,\dots,N} = (\bar{u}_i, \bar{v}_i)_{i=1,\dots,N}$ is a $2 \times n$ matrix with coefficients

$$(\bar{u}_i, \bar{v}_i) = \sum_{j\in\mathcal{N}_i,\ j\in\mathcal{B}} -\frac{D_{ij}}{\displaystyle\sum_{k\in\mathcal{N}_i} D_{ik}}(u_j, v_j) \tag{4}$$

As we have just presented, considering a triangular mesh as a mass spring model with constrained boundaries, naturally leads to solve linear systems. For this reason, such kind of methods are generally referred as *linear parameterizations with fixed boundaries*.

2.2 Barycentric Weights

Now the main difference between all these methods reside in the choice of the *spring constants*, *i.e. barycentric weights*. We provide here some of the possible choices given in the literature:

1. *Graph theory based* [15, 16]
$$D_{ij} = 1 \tag{5}$$
2. *Chord Length* []
$$D_{ij} = \frac{1}{\|\mathrm{p}_i - \mathrm{p}_j\|^2} \tag{6}$$
3. *Discrete Conformal* [13]
$$D_{ij} = \cot \alpha_{ij} + \cot \beta_{ij} \tag{7}$$
4. *Discrete Authalic* [3]
$$D_{ij} = \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{\|\mathrm{p}_i - \mathrm{p}_j\|^2} \tag{8}$$
5. *Intrinsic* [3]
$$D_{ij} = \mu * \frac{\cot \gamma_{ij} + \cot \delta_{ij}}{\|\mathrm{p}_i - \mathrm{p}_j\|^2} + (1-\mu) * \cot \alpha_{ij} + \cot \beta_{ij} \tag{9}$$

2.3 Boundary Mapping

As presented, the first step of linear parameterizations with fixed boundary is to choose the image of the boundary points in the parametric space.

The most common strategy is to fix the boundary points on a convex shape. Indeed it guarantees the bijectivity of the parameterization if positive barycentric weights are used. For these reasons, most of linear parameterization with fixed boundary take a square, or a circle as parameter domain.

2.4 Numerical Issues

As we mention previously in section 2.1, linear parameterizations with fixed boundary require to solve sparse linear systems. There are many ways to solve sparse linear sytems $A\mathrm{x} = \mathrm{b}$, but all methods can be classified into two categories:

iterative solvers starting from an initialized solution x^0, the solution is iteratively updated until convergence to the solution x. The difference between these methods reside in the way to update the solution x^i.

direct solvers factorize the matrix A into a product of matrices that are simple to invert. For example, a LU decomposition consists in finding the lower triangular matrix L and the upper triangular matrix[1] U such that the product LU is equal to A of the system to be solved. Then solving the linear system is equivalent to solve easy triangular systems, *i.e.*

$$A\mathrm{x} = \mathrm{b} \Leftrightarrow LU\mathrm{x} = \mathrm{b} \rightarrow \begin{cases} L \cdot \mathrm{y} = \mathrm{b} \\ U \cdot \mathrm{x} = \mathrm{y} \end{cases}$$

A nice study has been done [2] that illustrates the advantages an inconvenients of the multitude of solvers available in the case of mesh processing. We decided that it would be better not to hardcode the choice of the solver in the code. First because that would mean hardcoding the usage of the vnl solver, the only solver available to ITK by default which unfortunatly happen to be the slowest and the the less stable of the solvers

[1] If A is symmetric, then $U = L^t$ and the factorization is referred as a *Cholesky* factorization.

we could experiment so far, second because whichever solver is faster today might not be the fastest in one month given how fast the research on that field seems to be recently. We took an a-la C-GAl approach where the code is templated over the solver, even if our implementation then differs from C-GAL in many ways.

3 Design

Following the previous section, the design of the code is straightforward, each functionality should be independent in order to make the use of the class as flexible as possible. The design of our class is finally given in Fig. 2.

Figure 2: Design of our implementation. The parameterization class takes as input a 3D Surface Mesh $\mathcal{M}$ with at least one boundary, and the choice of the boundary mapping and the barycentric weights is made through the corresponding templates. It solves the rsulting sparse linear sytems using the solver given as a template. Its outputs a planar Mesh which is the mapping of $\mathcal{M}$ onto the parametric domain defined by the boundary mapping, given the barycentric weights.

3.1 Border Transform

We create an abstract class `itk::MeshBorderTransform` which defines the API of the border transform. All derived class must contain one method to provide a border map, and the boundary mapping from the 3D space onto the parametric domain, *i.e.* a 2D domain.

We provide two implementations of this abstract class for a square shaped domain and a a circle shaped domain.

3.2 Barycentric weights

For the bayrcentric weights, we define an abstract functor `itk::MatrixCoefficients`. It takes as input a 3D triangular mesh and one hald edge and return the barycentric weight corresponding to the given edge. All the existing functors are defined in `itkQEMeshParamMatrixCoeficients.h`:

- Graph theory based `class OnesMatrixCoefficients`
- Chord Length `class InverseEuclideanDistanceMatrixCoefficients`
- Discrete Conformal `class ConformalMatrixCoefficients`
- Discrete Authalic `class AuthalicMatrixCoefficients`
- Intrinsic `class IntrinsicMatrixCoefficients`
- Harmonic `class HarmonicMatrixCoefficients`

3.3 Parameterization

The parameterization itself is just a mater of building the sparse systems depending on the connectivity of the surface mesh and the barycentric weights. It will delegates solving the matris to the solver specified in the template, and then rebuild a mesh from the solution given by the solver.

3.4 Solving sparse linear system

Only a few parameters need to be modified in the solver. The solvers stops when the residual error is lower than a threshold or when the maximum number of iteration is reached. If the result is not satisfactory, there is great chance that the number of iterations was not set high enough. You can set a new one using `SetNumberOfIterations()`.

4 Software Requirements

You need to have the following software installed:

- Insight Toolkit 3.4 (with Review enabled)
- CMake 2.4

For the sake of simplicity, we create some specific traits for solving sparse linear systems. One can either use the one we wrote, or write this own one (if you want to use any other sparse direct solver for example) and follow the same principle.

4.1 Solvers

vnl being included in ITK, we provide a default trait for vnl. It is strongly suggested that the user use other solvers to have an interactive experience with parameterization. VNL should only be used as a proof-of-concept or if your input mesh is relatively small. We recommand TAUCS, for which we provide traits. Both should give you the same results or better with an increase of an order of magnitude in speed. In our experiments, it also seems that it is numerically more stable than vnl for ill-conditionned matrices.

Download and install TAUCS from the following page: http://www.tau.ac.il/~stoledo/taucs/

5 Example

The example given along this paper use `itkVTKPolyDataReader` for reading its input. If you don't have a mesh to test with, you can find many here: http://www.aim-at-shape.net By default, if several borders are present, the longer one is chosen. If run without arguments, the example displays its usage.

6 Future work

We hope that the code will make it into the ITK toolkit. Even if the code is clean and commented, it does not completely follows ITK developpement guide. For exemple we did not check the style using KWStyle; a file can contain several classes; the naming convention is not always respected, ... It's all minor and the API should remain stable which motivated our early submission to the IJ. The parameterization package open the way to remeshing techniques, and to shape matching algorithms that we would like to evaluate. itkQuadEdgeMesh also allows a lot of mesh processing algorithm to be implemented in an elegant and optimized fashion. We are planning to implement more and more of those starting with some decimation and smoothing algorithms.

References

[1] S Angenent, S Haker, A Tannenbaum, and R Kikinis. Conformal geometry and brain flattening. In *the 2nd International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 271–278, 1999.

[2] Mario Botsch, David Bommes, and Leif Kobbelt. Efficient linear system solvers for mesh processing. In *Invited paper at XIth IMA Conference on the Mathematics of Surfaces*, 2005.

[3] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21:209–218, 2002. Eurographics conference proceedings.

[4] 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.

[5] Y Gao, J Melonakos, and A Tannenbaum. Conformal flattening itk filter. In *Insight Journal - 2006 MICCAI Open Science Workshop*, 2006.

[6] A Gouaillard, L Florez-Valencia, and E Boix. itkquadedgemesh: A discrete orientable 2-manifold data structure for image processing. In *Insight Journal - July - december*, 2006.

[7] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S-T. Yau. Genus zero surface conformal mapping and its application to brain surface mapping. *IEEE Transaction on Medical Imaging*, 23(8):949–958, 2004.

[8] S Haker, S Angenent, A Tannenbaum, and R Kikinis. Nondistorting flattening for virtual colonoscopy. In *the 3rd International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 358–366, 2000.

[9] S Haker; S Angenent; A Tannenbaum; R Kikinis; G Sapiro; M Halle. Conformal surface parameterization for texture mapping. In *IEEE Transaction on Visualization and Computer Graphics*, pages 181–189, 2000.

[10] W. Hong, X. Gu, F. Qiu, M. Jin, and A. Kaufman. Conformal virtual colon flattening. In *Solid and Physics Modeling*, 2006.

[11] K. Hormann, B. Lévy, and A. Sheffer. Mesh parameterization: Theory and practice. In *SIGGRAPH 2007 Course Notes*, volume 2, pages 1–122, San Diego, CA, August 2007. ACM Press.

[12] Yu P, Grant P, Qi Y, Han X, Sgonne F, Pienaar R, Busa E, Pacheco J, Makris N, Buckner R, Golland P, and Fischl B. Cortical surface shape analysis based on spherical wavelets. *IEEE Transaction on Medical Imaging*, 26(4):582–597, 2007.

[13] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993.

[14] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, 2006.

[15] W. T. Tutte. Convex representations of graphs. In *Proceedings of the London Mathematical Society*, volume 10, pages 304–320, 1960.

[16] W. T. Tutte. How to draw a graph. In *Proceedings of the London Mathematical Society*, volume 13, pages 743–767, 1963.