Laplace Beltrami Filter on QuadEdge Meshes

Michael Bowers, Dr. Laurent Younes

October 11, 2011

Center for Imaging Science
Johns Hopkins University

Abstract

This document describes a contribution to the Insight Toolkit intended to support the process of statistical analysis in Computational Anatomy. The methods included here operate on open or closed triagulated surfaces (represented by a QuadEdgeMesh). The filter assigns basis function values as Point Data on each vertex of the Mesh.

This paper is accompanied with the source code, input data, parameters and output data that we used for validating the algorithm described in this paper. This adheres to the fundamental principle that scientific publications must facilitate **reproducibility** of the reported results.

Contents

1 Introduction

In the field of Computational Anatomy the shape of biological structures are compared mathematically between subjects. The difference or similarity in shape between subject structures is quantified and correlations between shape and disease can be produced and examined for statistical significance.

It is desirable to reduce the dimensionality of statistical comparisons between subjects, but maintain information that might be valuable in finding correlations between shape and pathology. A surface mesh can be represented in a way that its N most significant surface harmonics can be compared with other subjects.

The purpose of this filter is to use the Laplace-Beltrami operator to determine surface harmonics in terms of PointData at each vertex. In the same way that a sound signal can be approximately characterized by a combination of its most significant frequency components, so can a surface be expressed as a combination of its surface harmonics. This filter determines the requested N most significant harmonics.

2 Overview

The filter described in this report operates on `itk::QuadEdgeMesh` data, and provides the requested basis function as PointData in a copy of the Input Mesh.

3 Algorithm For Approximation of the Laplacian on Triangulated Surfaces

The basic algorithm implemented in this filter is based on [Qiu2006] and is similar to [Levy2009]. The algorithm visits all the faces of the triangulated mesh, determining each face's area and the areas of the associated vertices, then computes the laplacian operator as a sparse matrix over the vertices.

Let S be a triangulated surface with faces $f \in F$ and vertices $v \in V$. Let ψ be a function defined on vertices. We want to define the laplacian of ψ.

First define the gradient, defined as a function indexed by *faces*. Let e_1, e_2, e_3 be three edges forming a face f (with the correct orientation). Let (v_1, v_2, v_3) be its vertices so that $e_1 = v_3 - v_2$, $e_2 = v_1 - v_3$ and $e_3 = v_2 - v_1$. Let $c = (e_1 + e_2 + e_3)/3$ be the center of the face.

Define the gradient $u = \nabla\psi(f)$ on the face by $u = \alpha_1 e_1 + \alpha_2 e_2$ such that $u \cdot (v_k - c) = \psi(v_k) - \psi(c)$ for $k = 1,2,3$, with $\psi(c) := (\psi(v_1) + \psi(v_2) + \psi(v_3))/3$. Since this implies that $u \cdot (v_k - v_l) = \psi(v_k) - \psi(v_l)$, this gives

$$\begin{aligned} \psi(v_3) - \psi(v_2) &= (\alpha_1 e_1 + \alpha_2 e_2) \cdot e_1 \\ \psi(v_1) - \psi(v_3) &= (\alpha_1 e_1 + \alpha_2 e_2) \cdot e_2 \end{aligned}$$

Let ψ_f be the column vector $[\psi(v_1), \psi(v_2), \psi(v_3)]^T$, M the 2 by 3 matrix

$$M = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \end{pmatrix}$$

and G_f the matrix

$$G_f = \begin{pmatrix} |e_1|^2 & e_1 \cdot e_2 \\ e_1 \cdot e_2 & |e_2|^2 \end{pmatrix}.$$

With this notation, the previous system is $M\psi_f = G_f\alpha$; this implies $|u|^2 = \alpha^T G_f \alpha = \psi_f^T M^T G_f^{-1} M \psi_f$.

First note that $\det G_f = |e_1|^2 |e_2|^2 - (e_1 \cdot e_2)^2 = (|e_1||e_2| \sin\theta_3)^2$ where θ_3 is the angle at v_3. It is therefore

equal to $4a(f)^2$ where $a(f)$ is the area of f. Now

$$\begin{aligned} M^T G_f^{-1} M &= \det(G_f)^{-1} \begin{pmatrix} 0 & 1 \\ -1 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} |e_2|^2 & -e_1 \cdot e_2 \\ -e_1 \cdot e_2 & |e_1|^2 \end{pmatrix} \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \end{pmatrix} \\ &= \det(G_f)^{-1} \begin{pmatrix} |e_1|^2 & e_1 \cdot e_2 & e_1 \cdot e_3 \\ e_1 \cdot e_2 & |e_2|^2 & e_2 \cdot e_3 \\ e_1 \cdot e_3 & e_2 \cdot e_3 & |e_3|^2 \end{pmatrix} \end{aligned}$$

after computation and using $e_3 = -e_1 - e_2$. Let Σ_f denote this last matrix. We can write the approximation:

$$\int_S |\nabla \psi(s)|^2 d\sigma(s) \simeq \sum_{f \in F} \psi_f^T M^T G_f^{-1} M \psi_f a(f) = \sum_{f \in F} \frac{\psi_f^T \Sigma_f \psi_f}{4a(f)}.$$

We want to identify the operator $\psi \mapsto \Delta\psi$ (the discrete Laplacian on S) such that

$$\sum_{f \in F} \frac{\psi_f^T \Sigma_f \psi_f}{4a(f)} = -\sum_{v \in V} \psi(v)(\Delta\psi)(v)a(v).$$

where $a(v)$ is the area attributed to vertex v, that can be defined by $a(v) = (1/3)\sum_{f:v \in f} a(f)$. We can write:

$$\sum_{f \in F} \frac{\psi_f^T \Sigma_f \psi_f}{4a(f)} = \frac{1}{4} \sum_{v \in V} \psi(v) \sum_{f:v \in f} (|e(v)|^2 \psi(v) + e(v) \cdot e(v') \psi(v') + e(v) \cdot e(v'') \psi(v''))/a(f)$$

where $e(v)$ is the edge opposed to v in face f and v' and v'' are the other two vertices in f. This means that one should define

$$\Delta\psi(v) = -\frac{1}{4a(v)} \sum_{f:v \in f} (|e(v)|^2 \psi(v) + e(v) \cdot e(v') \psi(v') + e(v) \cdot e(v'') \psi(v''))/a(f)$$

One can rewrite this discrete Laplacian in terms of angles. For a vertex v and a face f such that $v \in f$, oriented as $f = (v, v', v'')$, let $\theta'_f(v)$ and $\theta''_f(v)$ denote the angles opposed to v in f ($\theta'_f(v)$ is the angle at v' and $\theta''_f(v)$ is the angle at v''). With this notation, one has (since $e_f(v) = v'' - v'$ and $e_f(v') = v - v''$)

$$e_f(v) \cdot e_f(v') = -\cos(\theta''_f(v)) |e_f(v)| \, |e_f(v')| = -2cot(\theta''_f(v))a(f)$$

Similarly, $e_f(v) \cdot e_f(v'') = -2cot(\theta'_f(v))a(f)$ and, since the sum of edges is 0,

$$|e_f(v)|^2 = -e_f(v) \cdot (e_f(v') + e_f(v'')) = 2(cot(\theta'(v)) + cot(\theta''(v)))a(f),$$

One can therefore write

$$\Delta\psi(v) = \frac{1}{2a(v)} \sum_{f:v \in f} (cot(\theta''_f(v))(\psi(v') - \psi(v)) + cot(\theta'_f(v))(\psi(v'') - \psi(v)))$$

These expressions allow us to write the Laplacian operator as a sparse matrix indexed over the vertices of the triangulated surface. Letting $V = (v_1, \ldots, v_N)$, define A to be a sparse matrix such that

$$\Delta\psi(v_k) = \sum_{l=1}^{N} A(k,l)\psi(v_l).$$

(A is sparse since $A(k,l) = 0$ if $k \neq l$ and v_k and v_l are not connected by an edge.) Besides the zeros, A has the following entries. We first assume that S has no boundary, so that each edge belongs to exactly two faces.

- If (k,l) is an edge, we can define β_{kl} and β'_{kl} to be the two angles opposed to the edge in the two faces that contain it. Then
$$A(k,l) = \frac{1}{2a(v_k)}(cot(\beta_{kl}) + cot(\beta'_{kl}))$$
- Then, $A(k,k) = -\sum_{l \neq k} A(k,l)$ i.e.,
$$A(k,k) = -\frac{1}{2a(v_k)} \sum_{l \sim k} (cot(\beta_{kl}) + cot(\beta'_{kl}))$$
where $l \sim k$ indicates that l and k are linked by an edge.

These definitions must be modified when surfaces have boundaries. The filter supports two types of boundary conditions: $\psi = 0$ on the boundary of S (Dirichlet boundary condition) or $\nabla\psi$ tangent to the boundary of S (von Neumann boundary condition).

For triangulated surfaces, Dirichlet boundary condition is directly implemented by restricting to non-boundary vertices. For such vertices, the formulae for $A(k,k)$ and $A(k,l)$ remain unchanged.

Von Neumann boundary condition can be discretized by adding dummy faces to the surfaces symmetrically to boundary edges. This results in an extended surface $\tilde{S}$ and a function ψ on S can be extended to $\tilde{S}$ by symmetry too.

The resulting new definition of $A(k,l)$ is identical to the one with closed surfaces if (k,l) is an interior edge. If (k,l) is a boundary edge, then there is only one β_{kl} and $A(k,l) = cot(\beta_{kl})/a(v_k)$ (so that the angle is counted twice). $A(k,k)$ is defined also in this case by
$$A(k,k) = -\sum_{l \sim k} A(k,l).$$
(Notice that this is not necessarily satisfied with the Dirichlet boundary condition.)

Letting D be the diagonal matrix with coefficients $D(k,k) = a(v_k)$, then $B = DA$ is a symmetric matrix (B is just defined like A without the normalizations nu $a(v_k)$). Finding the eigenvalues and eigenvectors of A is equivalent to solving the generalized eigenvalue problem
$$B\psi = \lambda D\psi.$$

Finally, notice that the definition
$$a(v) = \frac{1}{3} \sum_{f:v \in f} a(f)$$
can be modified as
$$a(v) = \sum_{f:v \in f} a_f(v)$$
where $a_f(v)$ is the area of the *Voronoï¿½ cell* of v in f (region in the face for which points are closer to v than to the other two vertices).

The expression of $a_f(v)$ depends on whether f is obtuse (has an angle larger than $\pi/2$) or not.

In the non-obtuse case, one has (with the same notation as before)
$$a_f(v) = \frac{1}{4}(|e_f(v')|^2 \cot(\theta'_f(v)) + |e_f(v'')|^2 \cot(\theta''_f(v))).$$

In the obtuse case, one must take $a_f(v) = a(f)/2$ if v is the obtuse angle and $a_f(v) = a(f)/4$ otherwise.

4 Implementation

This filter derives from itk::QuadEdgeMeshToQuadEdgeMeshFilter. The input consists of a triangulated itk::QuadEdgeMesh. Filter Data Generation code calls CopyInputMeshToOutputMesh so the output of the filter is structurally a copy of the input filter, of type TOutputMesh. The filter uses vnl_sparse_matrix to contain the Laplacian operator over the vertices. To compute eigenvalues and eigenvectors, the filter uses vnl_sparse_symmetric_eigensystem.

5 Usage

This filter derives from itk::QuadEdgeMeshToQuadEdgeMeshFilter, so the user needs to specify TInputMesh and TOutputMesh to instantiate the class. Some sample definitions:

```
typedef float             PixelType;
typedef double            PointDataType;
typedef double            DDataType;
typedef double            CoordRep;
typedef double            InterpRep;
const   unsigned int      Dimension = 3;

// Declare the type of the input and output mesh
typedef itk::QuadEdgeMeshTraits<PixelType, Dimension, PointDataType,
    DDataType, CoordRep, InterpRep> MeshTraits;
typedef itk::QuadEdgeMesh<float,Dimension,MeshTraits> InMeshType;
typedef itk::QuadEdgeMesh<double,Dimension,MeshTraits> OutMeshType;

typedef itk::LaplaceBeltramiFilter< InMeshType, OutMeshType >
                                        LbFilterType;
LbFilterType::Pointer lbFilter = LbFilterType::New();
```

The input should be a triangulated itk::QuadEdgeMesh based object. Meshes with polys of more than 3 sides will generate an exception in Update(). The user should specify the number of surface harmonics (lbFilter->SetEigenValueCount(eCount)) to generate before updating the filter. Once the filter data is generated (lbFilter->Update()), the user has access to:

- GetLBOperator - the sparse matrix of the Laplace operator over the vertices
- GetHarmonics - values for all the requested surface harmonics over all the vertices
- GetEigenvalues - values for all the eigenvalues of the solution
- SetSurfaceHarmonic - set the point data of the output filter to the values of the desired surface harmonic. Call GetOutput() to access this mesh.

6 Results

The results in this example can be obtained by executing the test program itkLaplaceBeltramiFilterTest2:

```
USAGE: itkLaplaceBeltramiFilterTest2 [OPTIONS] <vtk_mesh_file> <first_harmonic_surface>
            -h --help : print this message
            -e --eigenvalueCount : number of principal eigenvalues to calculate
```

The program will determine the Laplacian Operator values for each matrix. If if the user specifies a harmonic count *N* with `-e N` or `--eigenvalueCount N`, the program will produce *N* .vtk files containing the original mesh with the PointData set to the first *N* surface harmonics. The following example computes the first nine surface harmonics on the input mesh `fvo.vtk`, a hippocampus surface.

```
./itkLaplaceBeltramiFilterTest2 --eigenvalueCount 9 fv0.vtk lbOutput.vtk
```

Figure 1 shows the first nine output surface harmonics displayed in CAWorks, a JHU Center for Imaging Science Paraview-based application.

7 Acknowledgements

Funding for development provided by NIH grants (R01-EB008171-01A1 and P41-RR015241).

References

[Qiu2006] Anqi Qiu, Dmitri Bitouk, Michael I. Miller, ”Smooth Functional and Structural Maps on the Neocortex via Orthonormal Bases of the Laplace-Beltrami Operator”, IEEE Trans. Med. Imaging, 25, 1296-1306, 2006. 3

[Levy2009] Bruno Levy, Hao (Richard) Zhang, ”Spectral Mesh Processing”, SIGGRAPH Asia 2009, Course 32 3

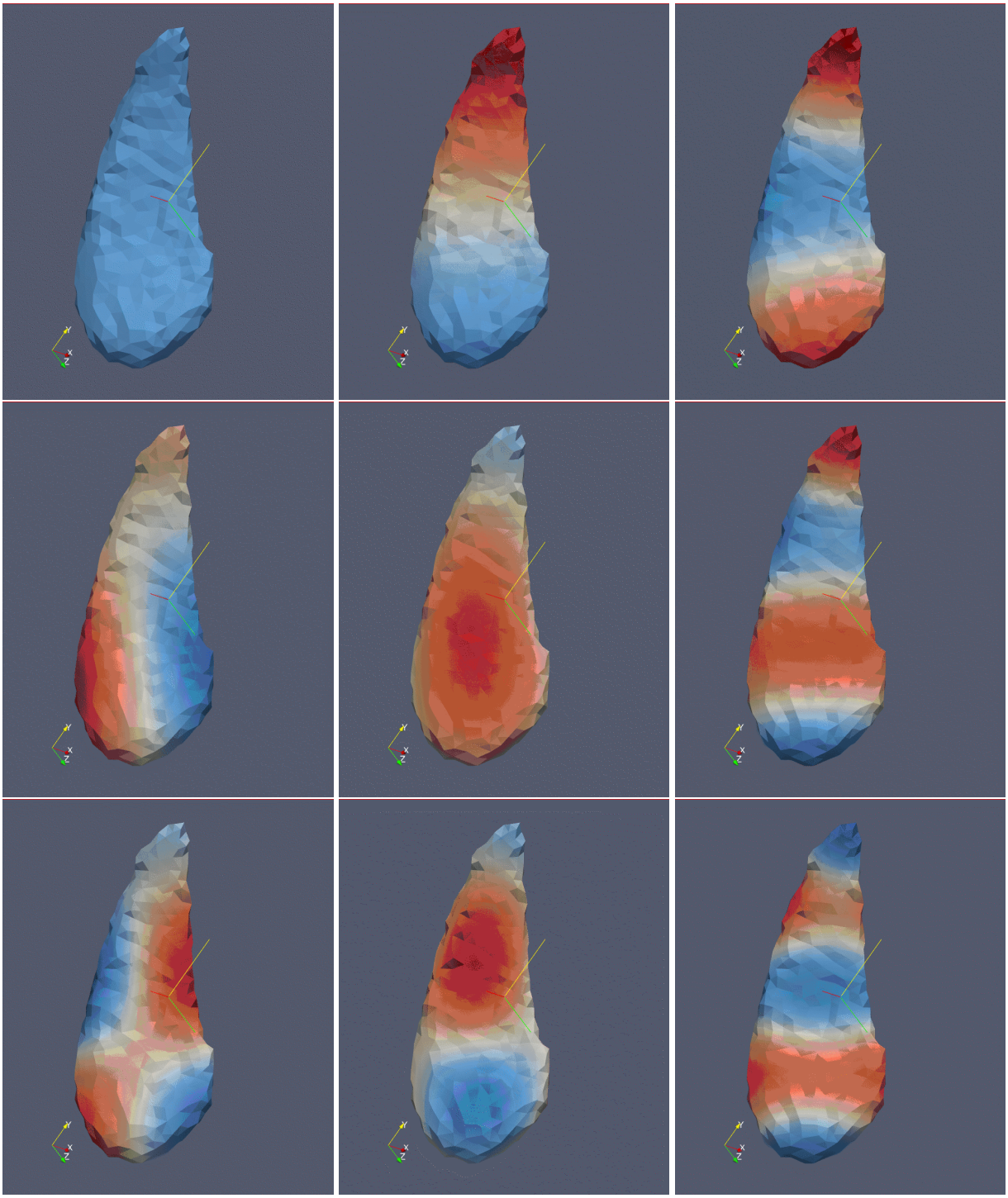

Figure 1: Results of running the Laplace-Beltrami operator on a triangulated hippocampus surface and calculating the first nine surface harmonics.