=1

Conformal Flattening ITK Filter

Release 0.00

Yi Gao[1], John Melonakos[1], and Allen Tannenbaum[1]

July 10, 2006

[1]Georgia Institute of Technology, Atlanta, GA

Abstract

This paper describes the Insight Toolkit (ITK) Conformal Flattening filter: itkConformalFlatteningFilter. This ITK filter is an implementation of a paper by Sigurd Angenent, et al., "On the Laplace-Beltrami Operator and Brain Surface Flattening" [1]. This filter performs an angle preserving map of any genus zero (i.e. no handles) triangulated mesh to the sphere or, alternatively, to the plane. In other words, any given triangle in the resulting sphere will have the same angles as the corresponding triangle in the original mesh, though their areas may differ. In this paper, we describe our code and provide the user with enough details to reproduce the results which we present in this paper. This filter has a variety of applications including the flattening of brain surfaces, which was the initial motivation for this work.

Contents

The folding and wrapping of brain surfaces presents researchers with difficulties in obtaining a more intuition-based surface rendering. In recent years, a number of methods have been proposed to map the brain surface to a plane or a sphere. Most of the previous methods are derived to preserve local area or length. In the paper of Sigurd Angenent, et al., "On the Laplace-Beltrami Operator and Brain Surface Flattening", an angle preserving conformal flattening scheme is proposed.

The algorithm obtains the explicit form of the flattening map by solving a partial differential equation on the surface using the finite element method (FEM). The mapping is a bijection and the original structure can be restored by inverse mapping.

1 Algorithm Details

1.1 Mathematical manipulation

Denote the genus zero surface which is to be flattened as Σ. Σ can be mapped to a plane using the algorithm proposed in [1]. The mapping is conformal thus the angles are preserved. Furthermore, the plane can be mapped to a sphere, using standard stereographic projection.

It is proven in the appendix of [1] that the mapping z, defined on Σ, satisfying is the following:

$$\triangle z = (\frac{\partial}{\partial u} - i\frac{\partial}{\partial v})\delta_p \tag{1}$$

where p is an (arbitrary) point on Σ. The funtion z maps $\Sigma\backslash\{p\}$ to the complex plane $\mathbf{C}$. Then by standard sterographic projection, the complex plane is mapped to a sphere excluding only the north pole.

1.2 Numerical scheme

The mapping z is defined on the surface. In the numerical manipulation where the surface is denoted as a triangulated mesh, solving of the equation (1) is carried out by FEM.

In the discrete configuration, z is approximated by a piecewise linear function on the triangulated mesh. Denote the finite-dimensional space $PL(\Sigma)$ of piecewise linear functions on Σ. To solve equation (1), first its right side should be approximated. For any function f smooth in a neighborhood of p,

$$\begin{aligned} & \int\int_\Sigma f(\frac{\partial}{\partial u} - i\frac{\partial}{\partial v})\delta_p \mathrm{d}S \\ = & -\int\int_\Sigma (\frac{\partial}{\partial u} - i\frac{\partial}{\partial v}) f\delta_p(w)\mathrm{d}S \\ = & -(\frac{\partial f}{\partial u} - i\frac{\partial f}{\partial v})\delta_p \end{aligned} \tag{2}$$

Thus $\forall f \in PL(\Sigma)$, given that the point p lies in the cell composed by three points A, B, and C, the quantity of (2) is determined by the values of f on A, B, and C.

On the triangle $\triangle ABC$, choose the u and v axes such that A and B are along u axis and the positive direction of axis v is pointing to the point C. So,

$$\begin{aligned} \frac{\partial f}{\partial u} &= \frac{f_B - f_A}{||B-A||} \\ \frac{\partial f}{\partial v} &= \frac{f_C - f_E}{||C-E||} \end{aligned}$$

in which E is the orthogonal projection of C on AB and can be computed by:

$$E = A + \theta(B-A)$$

where

$$\theta = \frac{< C-A, B-A >}{||B-A||^2}$$

in which $<,>$ denotes inner product.

Thus $\forall f \in PL(\Sigma)$,

$$(\frac{\partial f}{\partial u} - i\frac{\partial f}{\partial v})\delta_p = \frac{f_A - f_B}{||B-A||} + i\frac{f_c - (f_A + \theta(f_B - f_A))}{||C-E||} \tag{3}$$

Next we are going to perform the finite element solution of the mapping in the function space $PL(\Sigma)$.

FEM theory indicates that the solver function $z = x + iy$ for equation (1) is the minimizer of the Dirichlet functional:

$$\mathbf{D}(z) := \frac{1}{2}\int\int_{\Sigma}[|\nabla z|^2 + 2z(\frac{\partial}{\partial u} - i\frac{\partial}{\partial v})\delta_p]\mathrm{d}S \tag{4}$$

z satisfies equation(1) if and only if $\forall$ smooth function f,

$$\int\int_{\Sigma}\nabla z \cdot \nabla f \mathrm{d}S = (\frac{\partial f}{\partial u} - i\frac{\partial f}{\partial v})|_p \tag{5}$$

To find the function in $PL(\Sigma)$ satisfying equation(5), we define a set of basis ϕ_P in the $PL(\Sigma)$ as,

$$\begin{aligned} \phi_P(P) &= 1 \\ \phi_P(Q) &= 0, \forall Q \neq P \\ \phi_P(P) &\quad is \quad linear \quad on \quad each \quad triangle \end{aligned} \tag{6}$$

Then function z can be expressed as a linear combination of the basis:

$$z = \sum_{P:\ vertex\ of\ \Sigma} z_P \phi_P \tag{7}$$

By this we convert the original problem into solving for the complex vector $\vec{z} = (z_P)$:

$$\sum_P z_P \int\int \nabla\phi_P \cdot \nabla\phi_Q \mathrm{d}S = \frac{\partial \phi_Q}{\partial u}(p) - i\frac{\partial \phi_Q}{\partial v}(p) \tag{8}$$

Denote matrix D_{PQ} as

$$D_{PQ} = \int\int \nabla\phi_P \cdot \nabla\phi_Q \mathrm{d}S \tag{9}$$

The off-diagonal elements of D can be computed by:

$$\{\mathbf{D}\}_{P,Q} = -\frac{1}{2}(ctg\angle R + ctg\angle S), \quad P \neq Q \tag{10}$$

The points are illustrated in figure (1).

Also, since

$$\sum_P D_P Q = \sum_P \int \nabla\phi_P \cdot \nabla\phi_Q = \int \nabla 1 \cdot \nabla\phi_Q = 0 \tag{11}$$

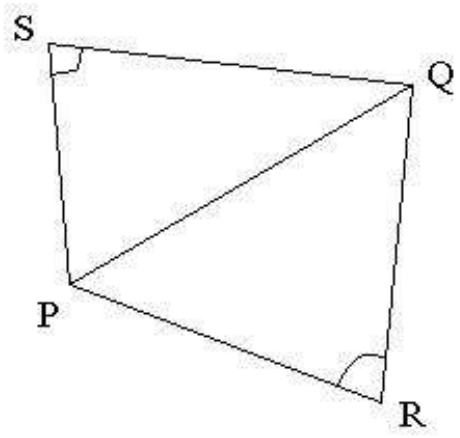


Figure 1: Triangle

Thus the diagonal elements of matrix D can be obtained by:

$$\mathbf{D}_{PP} = -\sum_{P \neq Q} D_P Q \tag{12}$$

Write each z_p as $z_p = x_p + iy_p$, then equation (8) can be written separately as:

$$\begin{aligned} \sum_P x_p D_{PQ} &= \frac{\partial \Phi(Q)}{\partial u}(P) \\ \sum_Q y_p D_{PQ} &= -\frac{\partial \Phi(Q)}{\partial v}(P) \end{aligned} \tag{13}$$

denote:

$$\begin{aligned} \vec{a} = (a_Q) &= \frac{\partial \Phi(Q)}{\partial u}(P) \\ \vec{b} = (b_Q) &= -\frac{\partial \Phi(Q)}{\partial v}(P) \end{aligned} \tag{14}$$

so we have:

$$\begin{aligned} \mathbf{D}\vec{x} &= \vec{a} \\ \mathbf{D}\vec{y} &= \vec{b} \end{aligned} \tag{15}$$

$\forall$ *point* Q:

$$a_Q - ib_Q := \begin{cases} 0 & for \quad Q \notin A,B,C \\ \frac{-1}{||B-A||} + i\frac{1-\theta}{||C-E||} & for \quad Q = A \\ \frac{1}{||B-A||} + i\frac{\theta}{||C-E||} & for \quad Q = B \\ i\frac{-1}{||C-E||} & for \quad Q = C \end{cases} \tag{16}$$

With vectors a, b, and matrix D, we can solve the linear equation to obtain the mapping. The matrix D is sparce, symmetric and positively defined, hence is suitable to being solved by the conjugate gradient method.

2 User's Guide

The conformal flattening filter takes an ITK mesh as input and will generate another ITK mesh as output. The usage is basically the same as other mesh to mesh filters in ITK.

2.1 Basic usage

The filter is instaintiated by:

```
typedef itk::ConformalFlatteningFilter< MeshType, MeshType>  FilterType;
FilterType::Pointer filter = FilterType::New();
```

Then the input can be set and results can be obtained by:

```
filter->SetInput( mesh );
```

and

```
newMesh = filter->GetOutput();
```

2.2 More about APIs

The filter has two categories of APIs for further manipulation. The first one is the setPointP function and the second contains two switch functions, mapToPlane and mapToSphere.

On the right side of equation (1), the δ_p function depands on the location of the point p. Basically, this point will be mapped to infinity on the plane and the north-pole of the sphere. Hence the selection of the point p determines which patch on the original mesh is mapped up to the north pole.

The API setPointP takes an integer as input indicating the cell number in which the point p lies. It's a good choice to set the point p where the local surface is relatively flat, i.e., having a small local curvature. However the computation of curvature can be done by the vtkCurvatures filter. In order to make this filter independent of VTK, this feature is not included in the filter and is left to the user. If setting the point p at some flat area is crucial, we suggest that user first use vtkCurvatures to obtain the number of cells having low curvatures and then call this function using one of the cells with a low curvature.

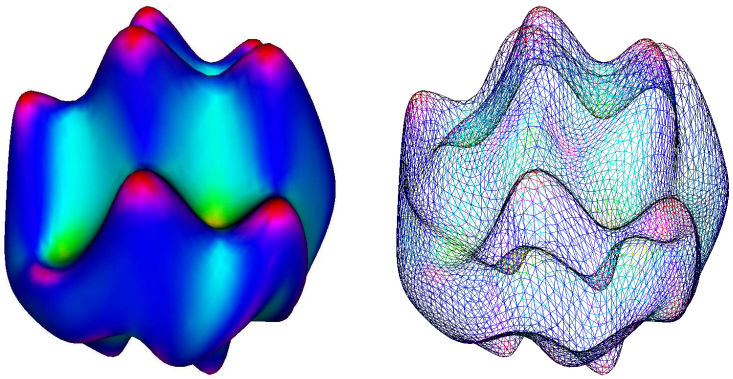
The switch functions mapToPlane and mapToSphere determine the output to be either a plane or a sphere, the sphere being the default. The difference between the two mappings is simply a stereographic projection from the plane to the sphere. Simply by

```
filter->mapToSphere( );
```

or

```
filter->mapToPlane( );
```

users can switch between two different outputs.



(a) Original Surface (b) Original Mesh

Figure 2: The Original Data

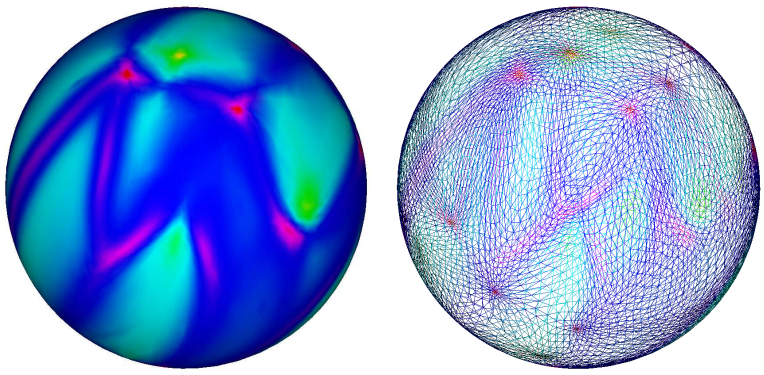
3 The Filter Test

This section contains details about the test we include with the submission of this ITK filter. We have provided *nice.vtk* which is a synthetic genus zero mesh. We set this as the single input to the itkConformalFlatteningFilterTest, as follows:

```
> ./itkConformalFlatteningFilterTest nice.vtk
```

In Figure 2, we show the original *nice.vtk* data, which is colored according to mean curvature. In Figure 2(a), we show the surface view and in Figure 2(b), we show the triangulated mesh.

In Figure 3, we show the results of the filter. The coloring provides a convenient visual mapping from the original mesh to the sphere. In Figure 3(a), we show the surface view of the sphere and in Figure 3(b), we show the triangulated mesh on the sphere.



(a) Flattened Surface (b) Flattened Mesh

Figure 3: The Results

In order to reproduce our results, the reader should compile and run our code with the following packages (though the code will likely compile and run with other variants, including CVS checkouts, of CMake, ITK, and VTK):

```
CMake 2.2.3
```

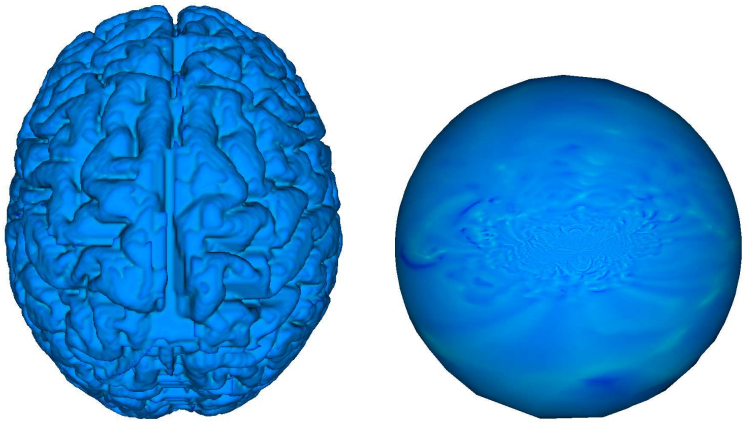
```
ITK 2.4.1
VTK 4.4.2
```

We have also included the results presented here as the *niceFlat.vtk* file included in this submission. One can verify the angle preserving nature of this filter by comparing ratios of edge lengths between triangles in the original surface and the corresponding triangles in the flattened mesh.

4 Future Applications

We propose to use this filter to conformally flatten brain surfaces. In order to accomplish this, we are currently investigating automated methods for extracting genus zero brain surfaces from raw MRI volumes.

We have tried a variety of brain extraction (skull-removing) tools, including FreeSurfer, BET, MRIcro, ITKSNAP, and other ITK levelset filters and have been able to extract brain surface meshes using these tools. However, thus far, we have been unsuccessful in obtaining a genus zero surface without significant manual editing, using these tools. We are hopeful that topology preserving levelset algorithms will soon become part of ITK to facilitate our work, but the development of this filter is beyond the scope of this paper. We are also aware of topology correction algorithms which perform cutting operations to convert any genus surfaces into genus zero surfaces. We will also investigate the use of these algorithms to extract genus zero brain surfaces.



(a) Original Brain Surface (b) Flattened Brain Surface

Figure 4: Preliminary Brain Surface Results

As described above, through the use of various open-source tools and a fair amount of manual editing, we were able to obtain a genus zero surface of the brain which we show in Figure 4(a). Preliminary results of running the itkConformalFlatteningFilterTest on *brain.vtk* can be seen in Figure 4(b). Note that the coloring is not optimal since the coloring parameters in the test filter have been optimized for the *nice.vtk* test file. In the future, we will optimize the coloring scheme for brain surfaces so that brain surface meshes mapped to the sphere will provide nice visual insights into the contours of brain sulci and gyri.

5 Conclusions

In this paper, we have described the Insight Toolkit (ITK) Conformal Flattening filter: itkConformalFlatteningFilter. This ITK filter is an implementation of a paper by Sigurd Angenent, et al., "On the Laplace-Beltrami Operator and Brain Surface Flattening" [1]. This filter performs an angle preserving map of any genus zero (i.e. no handles) triangulated mesh to the sphere or, alternatively, to the plane.

We have provide the user with details to be able to reproduce our results. We have also given suggestions as to how to exploit the full functionality of this filter.

In the future, we propose to use this filter to map entire brain surfaces to the sphere. Perhaps clinical studies of the resulting flattened map with yield insights into brain anatomy and function.

References

[1] S. Angenent, S. Haker, A. Tannenbaum, and R. Kikinis. On the Laplace-Beltrami operator and brain surface flattening. *Medical Imaging, IEEE Transactions on*, 18(8):700–711, 1999. (document), 1.1, 5