=1

Conformal Flattening ITK Filter

Release 0.00

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

July 22, 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) surface to the sphere or, alternatively, to the plane. 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. Like in the work of [3] and [4], a parameterized deformable surface whose topology is mappable to a sphere is fitted. Thus the brain surface can be represented on a planar map using spherical coordinates. Also, as in [2] and [5], quasi-isometrics and quasi-conformal flattening schemes are used to flatten the brain

surface. Those methods are more functional minimizing schemes and hence bijectivity can not be guaranteed. In the paper [1], an bijective angle preserving conformal flattening scheme is proposed. See the end of section 4 for a definition of angle preserving.

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 thus the triangles do not flip 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:

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

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).

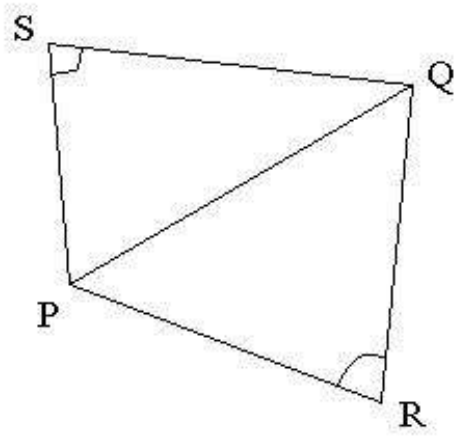


Figure 1: Triangle

Also, since

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

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

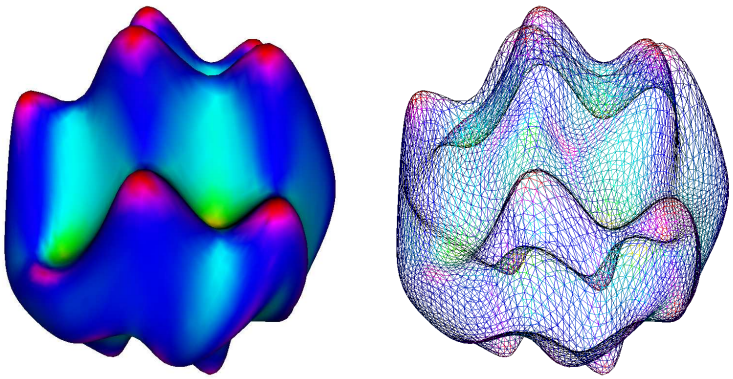
The filter has several APIs for further manipulation of the output.

1. setPointP function. 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.

2. 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( );
   ```



(a) Original Surface (b) Original Mesh

Figure 2: The Original Data

or

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

users can switch between two different outputs.

3. setScale function. The mapping, calculated from the equation (1), is dependent on the number of the nodes within the mesh. Given a mesh of a large number of nodes and cells, the image of the flattening mapping, is constrained in a small range around origin. To make it cover a sufficient area of the plane and further get a reasonable result from stereographic projection, re-scale of the flattened plane is needed. This function is used to set the scale factor, by:

   ```
   filter->setScale( scale factor );
   ```

   For a mesh of around several thousands of nodes, a factor of 100 is likely to get a good result. The factor should grow as the number of nodes grows.

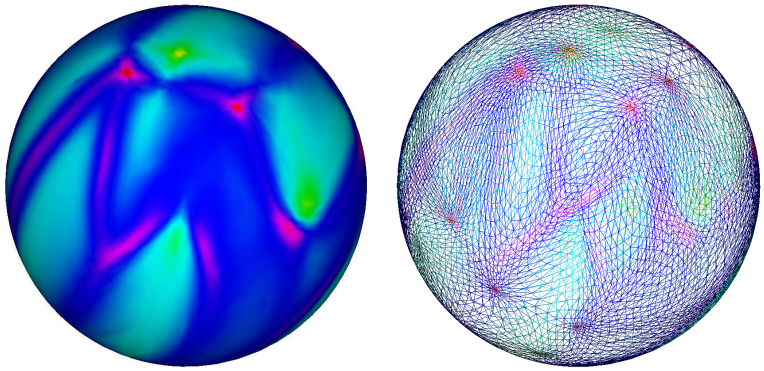
3 The Filter Test

Here we explain how the reader may reproduce our results. 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 first parameter to the itkConformalFlatteningFilterTest and we set the output filename for the flattend mesh as the second parameter, as follows:

```
> ./itkConformalFlatteningFilterTest nice.vtk niceFlat.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 conformally flattened result 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
```

Note that this code is optimized for this particular *nice.vtk* file and is not intended to be used with any generic mesh. There are two unique optimizations in this case:

1. The choice of the filter's scale parameter (using the setScale function. In this case, we chose: scale = 100.
2. The choice of rendering options to fit the scale of the mean curvature values (see our Display function in the itkConformalFlatteningFilterTest.cxx file).

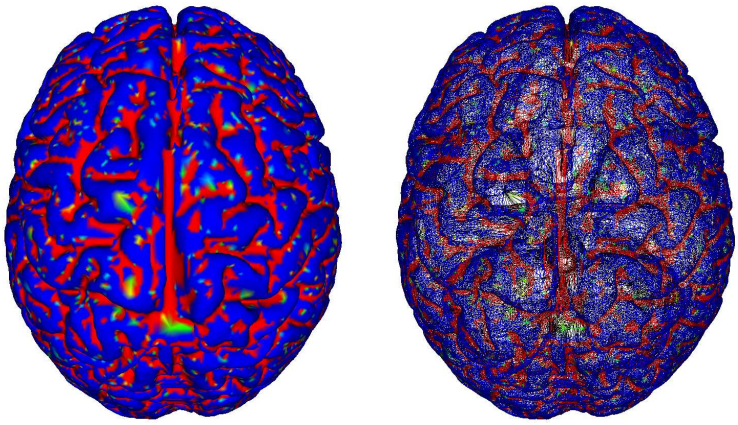
In the next section, we show code that is optimized for a brain surface dataset.

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 the relative angle proportions of angles emanating from each vertex in the mesh. We have included the anglePreserveCheck.cxx/h files to enable the user to verify that the angles are indeed preserved. We expond more on this in the next section.

4 Brain Flattening 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

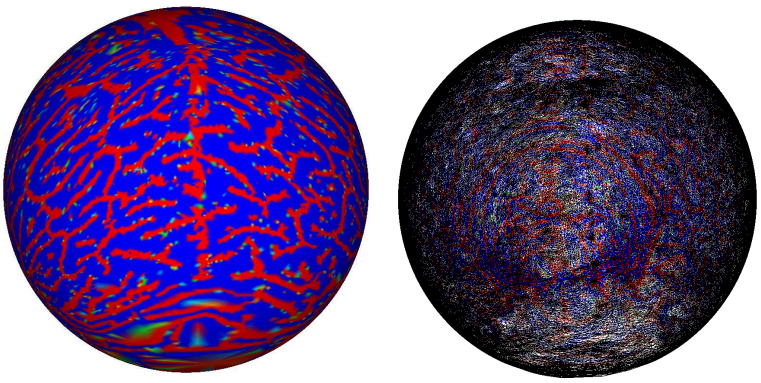


(a) Original Brain Surface (b) Original Brain Mesh

Figure 4: Original Brain Surface

editing, using these tools. We are hopeful that topology preserving levelset algorithms will soon become part of ITK to facilitate our work. 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 particular genus surface into a genus zero surface. We will also investigate the use of these algorithms to extract genus zero brain surfaces.

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 Figures 4(a) and 4(b). The result of running the brainTest.cxx on *brain.vtk* can be seen in Figures 5(a) and 5(b). We include the output file in this submission as $brainFlat.vtk$.



(a) Flattened Brain Surface (b) Flattened Brain Mesh

Figure 5: Flattened Brain Surface

We made the following optimizations in brainTest.cxx to achieve these results:

1. The choice of the filter's scale parameter (using the setScale function. In this case, we chose: scale = 5000.

2. The choice of rendering options to fit the scale of the mean curvature values (see our Display function in the brainTest.cxx file).

The mapping is conformal thus preserves angles. Since the surface is triangulated, the angle preserving property preserves the relative angles emanating from any given vertex in the mesh. Take, for example, any vertex P, which is mapped to point P'. There are M angles emanating from P, denoted as $A_1, A_2, \ldots, A_M$ for the original mesh and $A'_1, A'_2, \ldots, A'_M$ for the flattened mesh. The angle preserving property claims that

$$\frac{A_k}{sum_{i=1}^{M} A_i} = \frac{A'_k}{sum_{i=1}^{M} A'_i}, \quad \forall k \tag{17}$$

Numerically, the ratio of the left to right sides of equation (17) will be close to 1.0. We provide code to calculate and write to file the ratio in equation (17) for every angle at every vertex in the mesh. The usage of this angle preserving check is:

```
> ./anglePreserveCheck originalSurface.vtk flattenedSurface.vtk
```

The computed ratios will be written to *angleRatioDiff.dat*.

For the example given in Figure 2(a), the mean of all the ratios is 1.0026 and the standard deviation is 0.0629, thus verifying the angle preserving nature of this mapping.

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.

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

[2] G. J. Carman, H. A. Drury, and D. C. Van Essen. Computational methods for reconstructing and unfolding the cerebral cortex. *Cereb. Cortex*, 5(6):506C517, 1995. (document)

[3] C. Davatzikos and R. N. Bryan. Using a deformable surface model to obtain a shape representation of the cortex. *Medical Imaging, IEEE Transactions on*, 15(6):785C795, 1996. (document)

[4] D. MacDonald, D. D. Avis, and A. C. Evans. Multiple surface identification and matching in magnetic resonance images. In R. Robb, editor, *Visualization Biomed. Comput.*, page 160C169. New York: SPIE, 1994. (document)

[5] E. Schwartz, A. Shaw, and E. Wolfson. A numerical solution to the generalized mapmakers problem: Flattening nonconvex polyhedral surfaces. *Pattern Anal. Machine Intell., IEEE Transactions on*, 11:1005C1008, 1989. (document)