Spherical Thin Plate Spline Mesh Filter

Release 1.00

Wen Li[1,2], Vincent A. Magnotta[1,2,3]

February 13, 2012

[1]Department of Radiology, The University of Iowa, Iowa City, IA 52242
[2]Department of Biomedical Engineering, The University of Iowa, Iowa City, IA 52242
[3]Department of Psychiatry, The University of Iowa, Iowa City, IA 52242

Abstract

This document describes a contribution to the Insight Toolkit intended to perform landmark-based registration on two meshes. The method implemented here is restricted to meshes with a spherical geometry and topology. Please refer Wahba's paper[1] for the mathematical details and Zou's paper[2] for the applications.

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.

Acknowledgements

This work was funded in part by NIH/NINDS award NS050568.

Contents

1 Introduction

The filter described in this paper is for the spherical thin plate spline registration. There are currently a couple of thin plate spline transforms available in the Insight Toolkit but those are for 2D and 3D images. This filter provides the method to implement the thin plate spline transform on the sphere with two sets of matching

landmarks. It guarantees that the source landmarks move to the target landmarks and the point close to the source landmark move to somewhere close to the target landmark. The displacements are calculated with spherical coordinates (r,θ, ϕ).

2 Spherical Thin Plate Spline Interpolation

Wahba extended thin plate splines on Euclidean d-space to splines on the sphere[1]. If we have two sets of landmarks, the source landmark P_i and the target landmark Q_i, $i = 1,2,...,n$, the spherical thin plate spline problem is solved by minimizing $J_2(u)$ subject to $u(P_i) = z_i, i = 1,2,...,n$. Here, $z_i = Q_i - P_i$ is the displacement from the source landmark to the target landmark. J_2 can be written as

$$J_2(u) = \int_0^{2\pi} \int_0^{\pi} (\Delta u(\theta,\phi))^2 sin\phi d\theta d\phi, \tag{1}$$

where $\theta \in [0,\pi]$ is in the latitude direction, $\phi \in [0,2\pi]$ is in the longitude direction, and Δ is the Laplace-Beltrami operater.

The solution of the problem is to get the displacement at any point $P(\theta,\phi)$ by the interpolation given as

$$u_n(P) = \sum_{i=1}^{n} c_i K(P,P_i) + d, \tag{2}$$

where

$$\begin{gathered}
\mathbf{c} = \mathbf{K_n^{-1}}\left[\mathbf{I} - \mathbf{T}(\mathbf{T'K_n^{-1}T})^{-1}\mathbf{T'K_n^{-1}}\right]\mathbf{z}, \\
d = (\mathbf{T'K_n^{-1}T})^{-1}\mathbf{T'K_n^{-1}z}, \\
(\mathbf{K}_n)_{ij} = K(P_i,P_j), \\
\mathbf{T} = \begin{pmatrix} 1, & ..., & 1 \end{pmatrix}' \\
\mathbf{z} = \begin{pmatrix} z_i, & ..., & z_n \end{pmatrix}'
\end{gathered}$$

in which $(\mathbf{K}_n)_{ij}$ is the (i,j)th element of the $n \times n$ matrix $\mathbf{K}_n$, $\mathbf{z}$ is the $n \times 2$ matrix which keeps the displacement $(\Delta\theta_i, \Delta\phi_i)$ for each pairs of landmarks $P_i(\theta,\phi)$ and $Q_i(\theta,\phi)$. $K(P_i,P_j)$ in the matrix $\mathbf{K}_n$ or $K(P,P_i)$ in Equation 2 can be calculated as

$$K(X,Y) = \frac{1}{4\pi}\int_0^1 log_{10}h(1-\frac{1}{h})(\frac{1}{\sqrt{1-2ha+h^2}} - 1)dh, \tag{3}$$

where $a = cos(\gamma(X,Y))$ and $\gamma(X,Y)$ is the angle between X and Y.

3 How to Use the Filter

This section illustrates the minimum operations required for running the filter. The code shown here is available in the code that accompanies this paper. You can download the entire set of files from the Insight Journal web site. The data and the landmark file for testing can be found in the Data directory of the uploaded code package.

The example is to illustrate how to extract landmarks from the input fixed mesh (source) and the input moving mesh (target). The input landmark file keeps point IDs on the meshes. There is another input mesh is where the deformation takes place. It produces the output of the deformed mesh for input mesh.

Distributed under Creative Commons Attribution License

- LandmarkIds

In order to use this filter we should include the header files for the Spherical Thin Plate Spline Mesh filter, the reader and writer types, and the IO stream to read the .txt file.

```
#include "itkQuadEdgeMesh.h"
#include "itkQuadEdgeMeshTraits.h"

#include "itkVTKPolyDataReader.h"
#include "itkQuadEdgeMeshScalarDataVTKPolyDataWriter.h"

#include "itkQuadEdgeMeshSphericalThinPlateSplineFilter.h"

#include <fstream>
```

The Scalar type associated with the nodes in the mesh, and the mesh dimension are defined in order to declare the Mesh type

```
    typedef float       MeshPixelType;
    const unsigned int Dimension = 3;

    typedef itk::QuadEdgeMesh< MeshPixelType, Dimension >   MeshType;
```

In order to read the input meshes you must declare reader types for all of the input meshes. Next, create an instance for each reader.

```
    typedef itk::VTKPolyDataReader< MeshType >     ReaderType;

    ReaderType::Pointer fixedReader = ReaderType::New();
    fixedReader->SetFileName( argv[1] );
    fixedReader->Update( );

    ReaderType::Pointer movingReader = ReaderType::New();
    movingReader->SetFileName( argv[2] );
    movingReader->Update( );

    ReaderType::Pointer fixedIC4Reader = ReaderType::New();
    fixedIC4Reader->SetFileName(argv[3]);
    fixedIC4Reader->Update();
```

Assign mesh pointers to the fixed mesh and the moving mesh, outputs of the readers.

```
    MeshType::Pointer  meshFixed  = fixedReader->GetOutput();
    MeshType::Pointer  meshMoving  = movingReader->GetOutput();
```

You declare the type of the spherical thin plate spline mesh filter and instantiate it.

Distributed under Creative Commons Attribution License

```
    typedef itk::QuadEdgeMeshSphericalThinPlateSplineFilter< MeshType, MeshType >
SphericalTPSType;

    SphericalTPSType::Pointer   tpsFilter  = SphericalTPSType::New();
```

You specify the sphere center and radius. All of the input meshes are required to share the same center and radius.

```
    double radius = atof( argv[4] );
    tpsFilter->SetSphereRadius(radius);

    SphericalTPSType::PointType center;
    center.Fill( 0.0 );
    tpsFilter->SetSphereCenter(center);
```

Next, you need to read the landmark file and create a set of source landmarks and a set of target landmarks for the filter. Please note that the set of landmarks can also be created by the landmark location (x,y,z). In that case, we don't need to read in the fixed mesh and the moving mesh anymore, and the landmark file will have point locations instead of point IDs in it.

```
    std::ifstream pointsFile;
    pointsFile.open( argv[5] );

    //Create source and target landmarks
    typedef MeshType::PointType LandmarkPointType;
    typedef SphericalTPSType::PointSetType LandmarkPointSetType;
    LandmarkPointSetType::Pointer sourceLandmarks = LandmarkPointSetType::New();
    LandmarkPointSetType::Pointer targetLandmarks = LandmarkPointSetType::New();

    MeshType::PointIdentifier sourceId;
    MeshType::PointIdentifier targetId;

    LandmarkPointType sourcePoint;
    LandmarkPointType targetPoint;

    unsigned int i = 0;

    pointsFile >> sourceId;
    pointsFile >> targetId;

    while( !pointsFile.fail() )
    {
        //get the point through point ID
        meshFixed->GetPoint(sourceId,&sourcePoint);
        sourceLandmarks->SetPoint( i, sourcePoint );

        meshMoving->GetPoint(targetId,&targetPoint);
        targetLandmarks->SetPoint( i, targetPoint );

        std::cout<<sourcePoint<<std::endl;
        std::cout<<targetPoint<<std::endl;
        i++;

```


Distributed under Creative Commons Attribution License

```
        pointsFile >> sourceId;
        pointsFile >> targetId;

    }

    pointsFile.close();
```

The input mesh that is expected to deform using the spherical thin plate spline transform, along with the source landmarks and the target landmarks are fed to the filter.

```
    tpsFilter->SetFixedMesh(fixedIC4Reader->GetOutput());
    tpsFilter->SetSourceLandmarks(sourceLandmarks);
    tpsFilter->SetTargetLandmarks(targetLandmarks);
    tpsFilter->Update();
```

Finally, the output of the filter can be passed to a writer.

```
    LandmarkPointSetType::Pointer targetLandmarks = LandmarkPointSetType::New();

    MeshType::PointIdentifier sourceId;
    MeshType::PointIdentifier targetId;

    LandmarkPointType sourcePoint;
```

The results in the following section were generated with calls similar to

```
LandmarkIds fixedMeshWithScalarsIC1.vtk movingMeshWithScalarsIC1.vtk\
 fixedMeshWithScalarsIC1 100.0 LandmarkIDsOnFixedIC1.txt\
 deformedFixedMeshWithScalarsIC1.vtk
```

4 Results

Figure 1 shows the input meshes and the output of the spherical thin plate spline mesh filter. All of the meshes have radius of 100.0 and are centered at the origin of coordinates. The scalar values of input fixed mesh are not involved in the calculation. They are shown with the meshes to help us identify the displacement of landmarks.

The source landmarks are chosen from the mesh on the left of Figure 1, and the target landmarks are chosen from the mesh in the center of Figure 1. The point shown with the highest scalar value on the left mesh is supposed to match with the point with the highest scalar value on the center mesh.

The left mesh is input to the spherical thin plate spline filter to let it calculate the displacement of each point on that mesh, using the spherical thin plate spline interpolation. The output mesh is shown as the mesh on the right of Figure 1.

Distributed under Creative Commons Attribution License

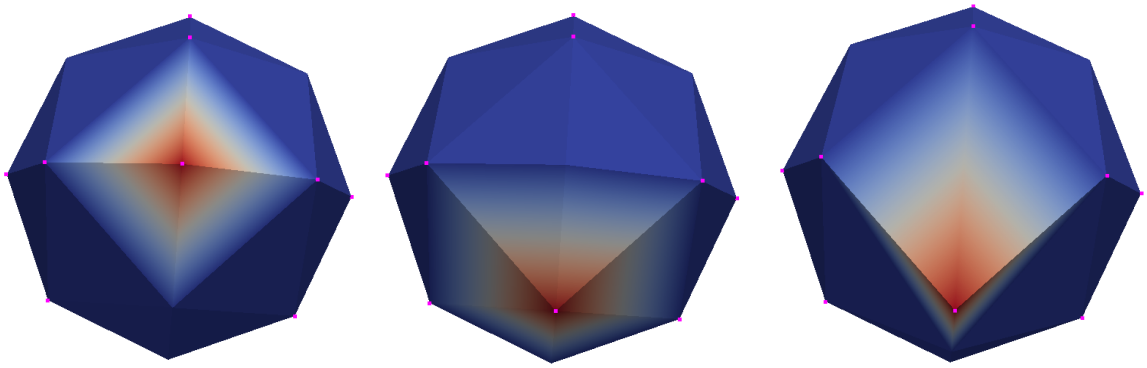

Figure 1: The Fixed Mesh (left), The Moving Mesh (center), and the Output Deformed Mesh (right).

Distributed under Creative Commons Attribution License

References

[1] G. Wahba. Spline interpolation and smoothing on the sphere. *SIAM Journal of Scientific and Statistical Computing*, 2(1):5–16, 1981. (document), 2

[2] G. Zou, J. Hua, and O. Muzik. Non-rigid surface registration using spherical thin-plate splines. In *Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention - Volume Part I*, MICCAI'07, pages 367–374, September 2007. (document)

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3201]
Distributed under Creative Commons Attribution License