Warp Mesh Filter

Release 1.00

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

August 25, 2010

[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 documents is about the filter itk::WarpQuadEdgeMeshFilter. It takes two input meshes (fixed and moving meshes) and one deformation field. It produces one output mesh. Each point on the fixed mesh is warped according to the displacement vector saved in the deformation field and gets a scalar value assigned by interpolating the scalar value from the moving mesh. An interpolator is needed to apply this filter.

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

This filter is needed after mesh registration to warp a mesh according to the deformation field produced by the registration. The deformation field can be generated from any kind of transform or the deformable registration. The deformation field is stored as a mesh with vector values associated with each vertex. For example, the vector values are displacements for points on fixed mesh in QuadEdgeMeshSphericalDiffeomorphicDemonsRegistration [2]. The filter takes three inputs in total.

Input 0: input fixed mesh.

Input 1: reference moving mesh.

Input 2: deformation field.

The deformation field and the input fixed mesh should have one to one point correspondence.

In QuadEdgeMeshSphericalDiffeomorphicDemonsRegistration [2], deformation field keeps the information of distance from the points on fixed mesh to the final destination. And the registered mesh could be the copy of the fixed mesh but with resampled scalar values, so the output of this filter is a copy of the input fixed mesh with resampled scalar values from the reference moving mesh by interpolation.

2 How to Build

This contribution includes

- Warp Mesh filter
- Tests for the filter
- All the LaTeX source files of this paper

The source code is in Source directory and named as

- itkWarpQuadEdgeMeshFilter.h
- itkWarpQuadEdgeMeshFilter.cxx

The testing code is in Testing directory as

- itkWarpQuadEdgeMeshFilterTest.cxx

2.1 Building Executables and Tests

In order to build the whole, it is enough to configure the directory with CMake. As usual, an out-of-source build is the recommended method.

In a Linux environment it should be enough to do the following:

- `ccmake SOURCE_DIRECTORY`
- `make`
- `ctest`

Where SOURCE_DIRECTORY is the directory where you have expanded the source code that accompanies this paper.

This will configure the project, build the executables, and run the tests and examples.

2.2 Building this Report

In order to build this report you can do

- `ccmake SOURCE_DIRECTORY`
- Turn ON the CMake variable
  - `BUILD_REPORTS`
- `make`

This should produce a PDF file in the binary directory, under the subdirectory `Documents/Report004`.

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 Testing directory of the code that accompanies this paper. You can download the entire set of files from the Insight Journal web site.

It is assumed that the input fixed mesh and the deformation field have one to one correspondence. An interpolator is needed to apply scalar values interpolation from the input moving mesh. It is set to be itk::LinearInterpolateMeshFunction [1] by default. The source code presented in this section can be found in the `Testing` directory under the filename.

- itkWarpQuadEdgeMeshFilterTest

In order to use this filter we should include the header files for the Warp Mesh filter, the reader and writer types and the interpolator.

```
#include "itkQuadEdgeMeshTraits.h"

#include "itkQuadEdgeMeshVTKPolyDataReader.h"
#include "itkQuadEdgeMeshScalarDataVTKPolyDataWriter.h"

#include "itkLinearInterpolateMeshFunction.h"
#include "itkWarpQuadEdgeMeshFilter.h"
```

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 >    InputMeshType;
typedef itk::QuadEdgeMesh< MeshPixelType, Dimension >    ReferenceMeshType;
```

The Vector type associated with the nodes in the deformation field, and the mesh dimension will be the same with the scalar meshes.

```
typedef InputMeshType::PointType   PointType;
typedef PointType::VectorType      VectorType;

typedef itk::QuadEdgeMeshTraits< VectorType, Dimension, bool, bool > VectorPointSetTraits;
typedef itk::QuadEdgeMesh< VectorType, Dimension, VectorPointSetTraits > MeshWithVectorsType;
```

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

```
typedef itk::QuadEdgeMeshVTKPolyDataReader< MeshWithVectorsType >
DeformationFieldReaderType;
typedef itk::QuadEdgeMeshVTKPolyDataReader< InputMeshType >          InputReaderType;
typedef itk::QuadEdgeMeshVTKPolyDataReader< ReferenceMeshType >
ReferenceReaderType;

InputReaderType::Pointer inputMeshReader = InputReaderType::New();
ReferenceReaderType::Pointer referenceMeshReader = ReferenceReaderType::New();
DeformationFieldReaderType::Pointer deformationFieldReader
    = DeformationFieldReaderType::New();
```

You declare the type of the warp mesh filter and instantiate it.

```
typedef itk::WarpQuadEdgeMeshFilter<
    InputMeshType, ReferenceMeshType, MeshWithVectorsType> WarpFilterType;

WarpFilterType::Pointer warpFilter = WarpFilterType::New();
```

You declare the type of the interpolator and instantiate it.

```
typedef itk::LinearInterpolateMeshFunction< ReferenceMeshType > InterpolatorType;

InterpolatorType::Pointer interpolator = InterpolatorType::New();
```

The outputs of the readers are passed as inputs to the warp mesh filter.

```
warpFilter->SetInputMesh (inputMeshReader->GetOutput());

warpFilter->SetReferenceMesh (referenceMeshReader->GetOutput());

warpFilter->SetDeformationField (deformationFieldReader->GetOutput());
```

And the interpolator is appointed to the warp mesh filter.

```
warpFilter->SetInterpolator(interpolator);
```

The execution of the filter can be triggered by calling the `Update()` method.

```
warpFilter->Update();
```

The output of the filter can be passed to a writer.

```
//write the result
typedef itk::QuadEdgeMeshScalarDataVTKPolyDataWriter< InputMeshType >  WriterType;
WriterType::Pointer writer = WriterType::New();
writer->SetInput( warpFilter->GetOutput());
writer->SetFileName(argv[4]);
writer->Update();
```

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

```
WarpQuadEdgeMesh fixedMesh.vtk movingMesh.vtk \
deformationField.vtk warpedMesh.vtk
```

4 Results

Figure 1 shows the input meshes and the output of the warp mesh filter. All of the meshes have radius of 100.0 units and are centered at the origin of coordinates. The scalar values of input fixed mesh are not involved in the warp filter, so they are not shown here. The scalar funtion on the surface of the input moving mesh is a Gaussian of the angle ϕ scaled by a constant of 2.0.

The deformation field mesh is computed from a result of registraion. It saves the distance from the points on fixed mesh to the destination points. The vectors on the mesh are shown with the arrows on it and the lengths of the arrows show the magnitude of the vectors.

The output mesh is the copy of the input fixed mesh but with interpolated scalars on it. Scalars are calculated in the input moving mesh after applying deformation field on the points of the input fixed mesh.

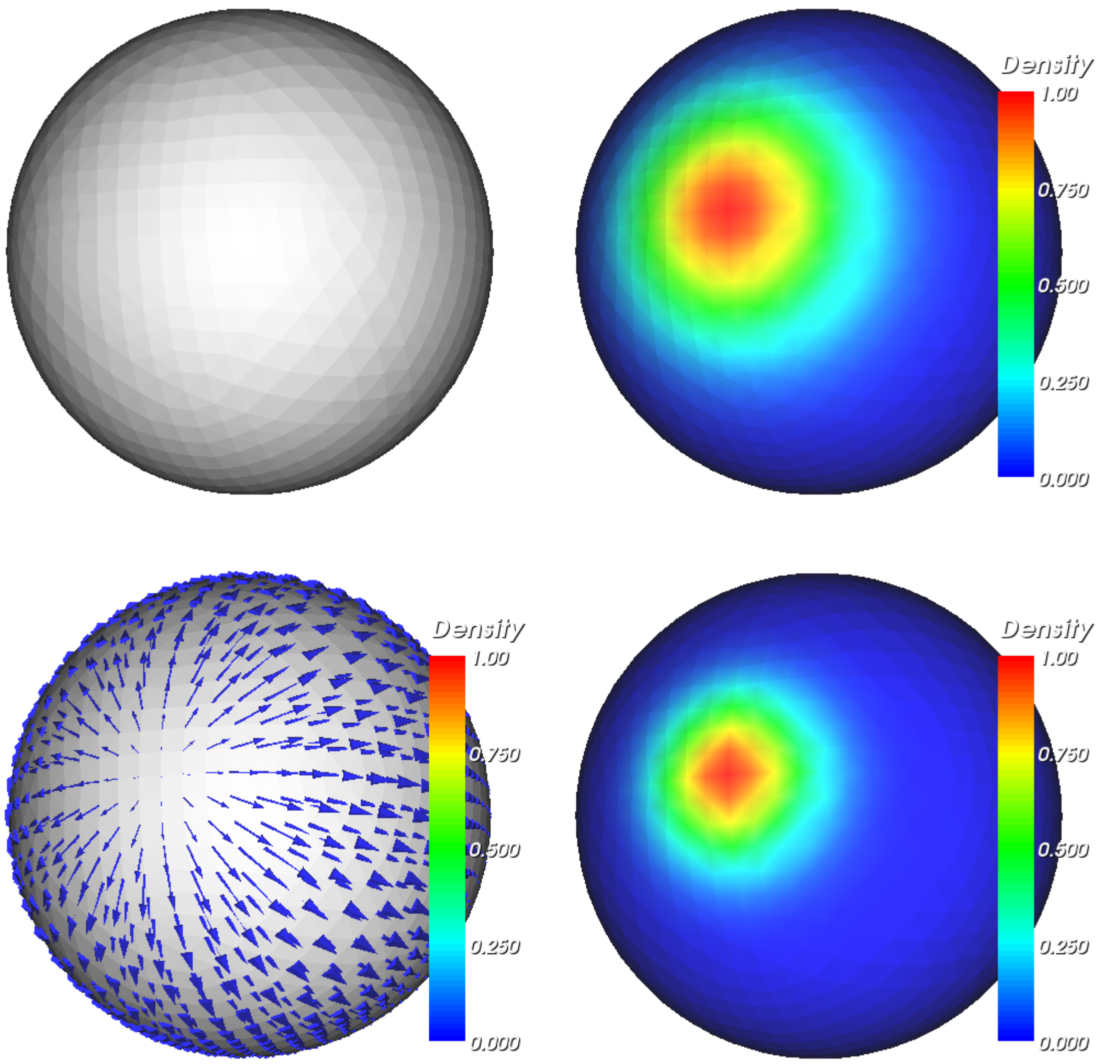

Figure 1: Input Fixed Mesh (upper left), Input Moving Mesh (upper right), Deformation Field (lower left) and Output Mesh (lower right) produced as output by the warp mesh filter.

References

[1] L. Ibanez, M. Audette, B. T. T. Yeo, and P. Golland. Rotational registration of spherical surfaces represented as quadedge meshes. *Insight Journal*, 2009. 3

[2] L. Ibanez, M. Audette, B. T. T. Yeo, and P. Golland. Spherical demons registration of spherical surfaces. *Insight Journal*, 2009. 1