Assign Scalars 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 article describes the filter itk::AssignScalarValuesQuadEdgeMeshFilter. It takes two meshes, one as an input and the other as a source to assign scalar values of the nodes from the source mesh to the input mesh. Both of the two meshes should have the same number of nodes.

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

2 How to Build

This contribution includes

- Assign Scalars filter
- Tests for the filter
- Examples on how to use the filter
- All the LaTeX source files of this paper

The source code is in Source directory and named as

- itkAssignScalarValuesQuadEdgeMeshFilter.h
- itkAssignScalarValuesQuadEdgeMeshFilter.cxx

The testing code is in Testing directory as

- itkAssignScalarValuesQuadEdgeMeshFilterTest1.cxx

The example about how to use it is in Example directory as

- RigidAndDemonsRegistration.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/Report003`.

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 mesh and the source mesh have the same number of nodes. The source code presented in this section can be found in the `Testing` directory under the filename.

- itkAssignScalarValuesQuadEdgeMeshFilterTest1

In order to use this filter we should include the header for the Assign Scalars filter, the reader and writer types and the `itk::QuadEdgeMesh` itself.

```
#include "itkAssignScalarValuesQuadEdgeMeshFilter.h"
#include "itkQuadEdgeMeshScalarDataVTKPolyDataWriter.h"
#include "itkQuadEdgeMeshVTKPolyDataReader.h"
#include "itkQuadEdgeMesh.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 >    SourceMeshType;
  typedef itk::QuadEdgeMesh< MeshPixelType, Dimension >    OutputMeshType;
```

You need to declare the type of the filter and instantiate it.

```
  typedef itk::AssignScalarValuesQuadEdgeMeshFilter<
                                   InputMeshType,
                                   SourceMeshType,
                                   OutputMeshType >     FilterType;

  FilterType::Pointer    filter   = FilterType::New();
```

In order to read the input mesh you need to declare the reader types for both the input and source meshes. You will want to create one instance for each reader.

```
  typedef itk::QuadEdgeMeshVTKPolyDataReader< InputMeshType >    InputReaderType;
  typedef itk::QuadEdgeMeshVTKPolyDataReader< SourceMeshType >   SourceReaderType;

  InputReaderType::Pointer inputReader = InputReaderType::New();
  inputReader->SetFileName( argv[1] );

  SourceReaderType::Pointer sourceReader = SourceReaderType::New();
  sourceReader->SetFileName( argv[2] );
```

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

```
filter->SetInputMesh(inputReader->GetOutput());
filter->SetSourceMesh(sourceReader->GetOutput());
```

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

```
filter->Update();
```

The output of the filter can be passed to a writer. The output mesh should be the same as the input mesh but with scalars from the source mesh.

```
typedef itk::QuadEdgeMeshScalarDataVTKPolyDataWriter< OutputMeshType >  WriterType;
WriterType::Pointer writer = WriterType::New();
writer->SetInput( filter->GetOutput() );
writer->SetFileName(argv[3]);
writer->Update();
```

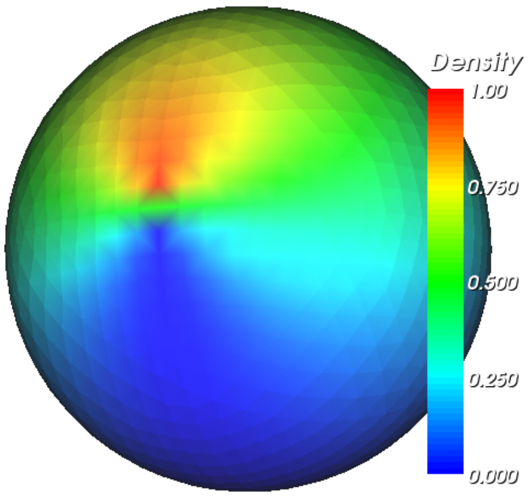
The results in the following section were generated with calls similar to

```
AssignScalarValuesQuadEdgeMesh fixedMesh01.vtk movingMesh01.vtk \
newMesh01.vtk
```

4 Results

Figure 1 shows the input mesh and the source mesh. Both of them have scalar values on them. By using this filter, the input mesh on the left is going to take the scalar values from the source mesh on the right to replace the original scalar values of its own.

Figure 2 shows the output mesh which is a copy of the input mesh but has the scalar values from the source mesh.



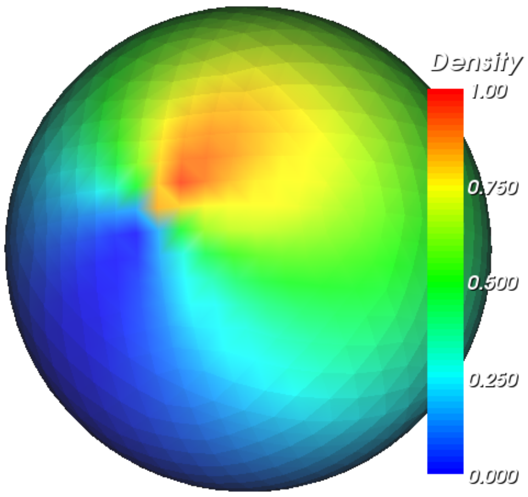


Figure 1: Input Mesh (left) and Source Mesh (right) passed as input to the Assign Values filter.

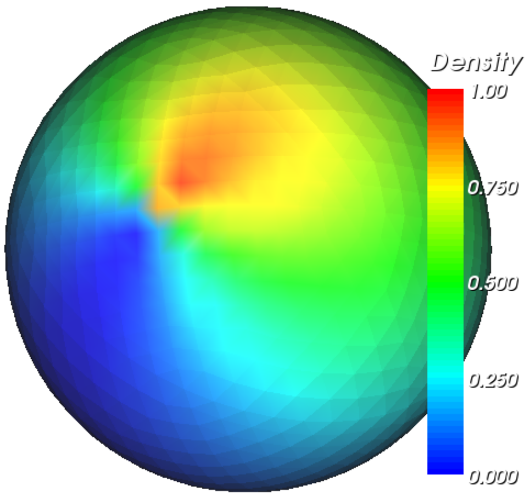


Figure 2: The output of the filter after assign scalar values from the source mesh to the input mesh.

References