Histogram Matching 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 describes the filter itk::HistogramMatchingQuadEdgeMeshFilter. It takes an input (source) mesh and a reference mesh, normalizes the scalar values between two meshes by histogram matching, and generates an output mesh which has a similar histogram as the input mesh.

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 HistogramMatchingQuadEdgeMeshFilter normalizes the scalar values of a source mesh based on the scalar values of a reference mesh. This filter uses a histogram matching technique where the histograms of the two meshes are matched only at a specified number of quantile values. This filter was originally designed to normalized scalar values of two brain surfaces.

No mean scalar value is used to remove the background as the itkHistogramMatchingImageFilter does [1, 2]. It is assumed that all mesh scalars are of interest and that the process of region definition and mesh generation have removed irrelevant information.

2 How to Build

This contribution includes

- Histogram Matching Mesh Filter
- Tests for the filter
- All the LaTeX source files of this paper

The source code is in Source directory and named as

- itkHistogramMatchingQuadEdgeMeshFilter.h
- itkHistogramMatchingQuadEdgeMeshFilter.txx

The testing code is in Testing directory as

- itkHistogramMatchingQuadEdgeMeshFilterTest.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/Report008`.

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.

The source code presented in this section can be found in the `Testing` directory under the filename.

- itkHistogramMatchingQuadEdgeMeshFilterTest

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

```

#include "itkQuadEdgeMesh.h"
#include "itkRescaleScalarsQuadEdgeMeshFilter.h"

#include "itkQuadEdgeMeshVTKPolyDataReader.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 >    MeshType;
```

We declare the type of the hisgtogram matching filter and instantiate it.

```
  typedef itk::HistogramMatchingQuadEdgeMeshFilter < MeshType, MeshType >
FilterType;

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

In order to read the input mesh we declare readers type for input and reference mesh and create instances of them.

```
typedef itk::QuadEdgeMeshVTKPolyDataReader< MeshType >   SourceReaderType;
typedef itk::QuadEdgeMeshVTKPolyDataReader< MeshType >   ReferenceReaderType;

SourceReaderType::Pointer srcReader = SourceReaderType::New();
srcReader->SetFileName( argv[1] );

ReferenceReaderType::Pointer refReader = ReferenceReaderType::New();
refReader->SetFileName( argv[2] );
```

The outputs of the readers are passed as inputs to the histogram matching filter. The source mesh can be set via either SetInput() or SetSourceMesh(). The reference image can be set via SetReferenceMesh().

```
filter->SetInput(srcReader->GetOutput());
filter->SetReferenceMesh(refReader->GetOutput());
```

SetNumberOfHistogramLevels() sets the number of bins used when creating histograms of the source and reference meshes. SetNumberOfMatchPoints() governs the number of quantile values to be matched.

```
filter->SetNumberOfHistogramLevels( atoi(argv[4]) );
filter->SetNumberOfMatchPoints( atoi(argv[5]) );
```

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.

```
typedef itk::QuadEdgeMeshScalarDataVTKPolyDataWriter< MeshType >  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

```
HistogramMatchingQuadEdgeMesh inputMesh.vtk referenceMesh.vtk \
outputMesh.vtk 1024 7
```

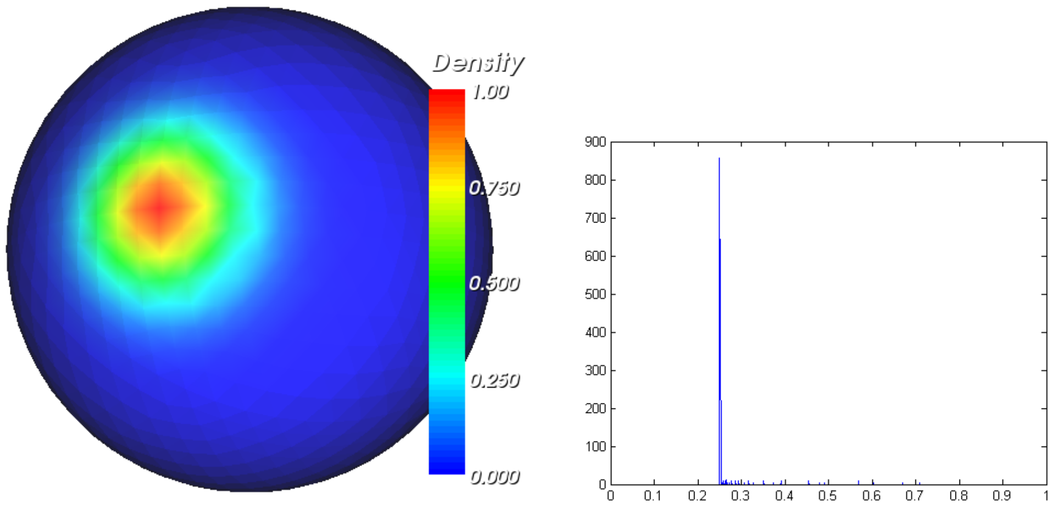

Figure 1: Input Mesh (left) and its histogram (right).

4 Results

Figure 1 shows the input mesh. The scalar values of it are in the range of $[0.0, 1.0]$ and the histogram of it is shown in the right.

Figure 2 shows the reference mesh. The scalar values of it are in the range of $[0.5, 1.0]$ and the histogram of it is shown in the right.

Figure 3 shows the output mesh which is a copy of the input mesh but has the scalar values in the range of $[0.5, 1.0]$, as the reference mesh and the histogram of it is shown on the right.

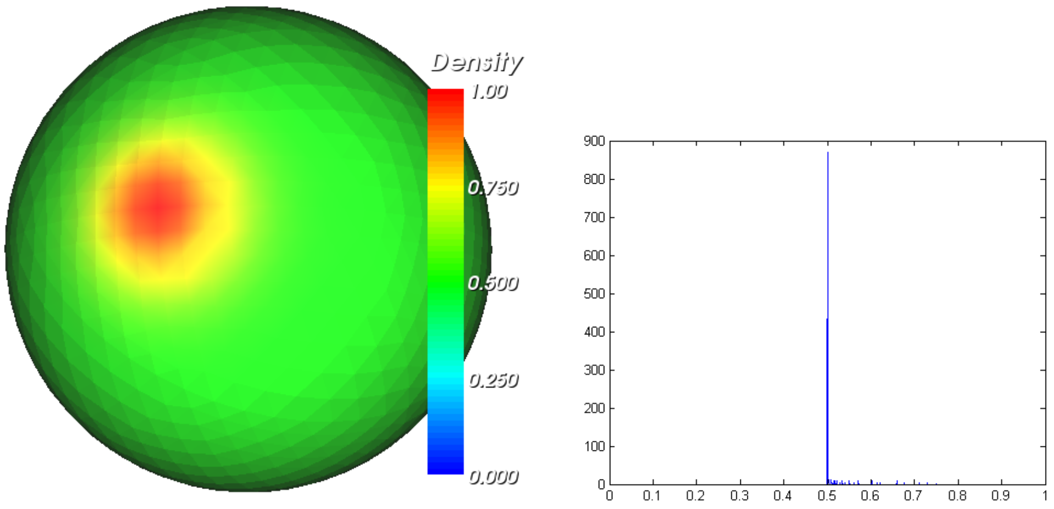

Figure 2: Reference Mesh (left) and its histogram (right).

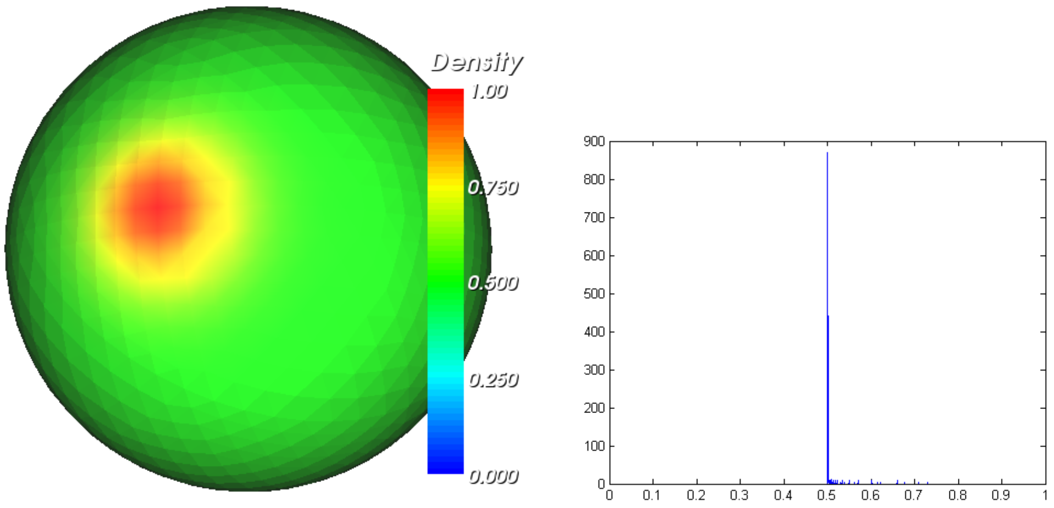

Figure 3: Output Mesh (left) and its histogram (right).

References

[1] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-10-6, http://www.itk.org/ItkSoftwareGuide.pdf, first edition, 2003. 1

[2] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, http://www.itk.org/ItkSoftwareGuide.pdf, second edition, 2005. 1