Mesh To List Adaptor

Release 1.00

Wen Li[1,2], Vincent A. Magnotta[1,2,3],Luis Ibanez[4]

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
[4]Kitware Inc., Clifton Park, NY

Abstract

This documents is about the filter itk::ScalarQuadEdgeMeshToListAdaptor. It takes the input mesh and generates a list of measurement vectors according to the scalars of the mesh. The list can be fed into `itk::Statistics::SampleToHistogramFilter` [1, 2] to generate a histogram about the scalars on 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

itkScalarQuadEdgeMeshToListAdaptor requires a mesh as input. Each point of the mesh could have an associated scalar value. This filter can only deal with those meshes with scalar pixel values, so the length of measurement vectors is fixed to be 1 and can not be changed by users.

The filter takes a mesh as input and goes through its `PointDataContainer` and push the scalars associated with the PointData into stack.

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

- itkScalarQuadEdgeMeshToListAdaptor.h
- itkScalarQuadEdgeMeshToListAdaptor.cxx

The testing code is in Examples directory as

- ScalarQuadEdgeMeshHistogram.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/Report006`.

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 `Examples` directory under the filename.

- ScalarQuadEdgeMeshHistogram

In order to use this filter you should include the header for the mesh to list adaptor, the reader and the `itk::QuadEdgeMesh` itself. And to show how to generate a histogram by using it, the header for `itkSampleToHistogramFilter` is included as well.

```
#include "itkVector.h"
#include "itkListSample.h"
#include "itkHistogram.h"
#include "itkQuadEdgeMesh.h"

#include "itkQuadEdgeMeshVTKPolyDataReader.h"
#include "itkScalarQuadEdgeMeshToListAdaptor.h"
#include "itkSampleToHistogramFilter.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;
```

In order to read the input meshes you declare reader type for the input mesh and create the instance of it.

```
typedef itk::QuadEdgeMeshVTKPolyDataReader< MeshType >          InputReaderType;

InputReaderType::Pointer inputMeshReader = InputReaderType::New();
```

You declare the type of the mesh to list adaptor and instantiate it.

```
typedef itk::ScalarQuadEdgeMeshToListAdaptor< MeshType >   AdaptorType;

AdaptorType::Pointer adaptor = AdaptorType::New();
```

The output of the reader is passed as input to the adaptor.

```
adaptor->SetMesh(  inputMeshReader->GetOutput() );
```

The execution of the filter can be triggered by calling the `Compute()` method. This should be done inside a `try/catch` block, since it is possible that error conditions may generate exceptions.

```
try
  {
adaptor->Compute();
  }
catch( itk::ExceptionObject & exp )
  {
  std::cerr << exp << std::endl;
  return EXIT_FAILURE;
  }
```

We declare the type of `itk::Statistics::SampleToHistogramFilter` and also the type of `ListSampleType` and `HistogramType` are declared. The instance of `itk::Statistics::SampleToHistogramFilter` is created.

```
typedef AdaptorType::MeasurementType  MeasurementType;
typedef AdaptorType::MeasurementVectorType  MeasurementVectorType;
typedef itk::Statistics::ListSample< MeasurementVectorType > ListSampleType;

typedef itk::Statistics::Histogram< MeasurementType,
        itk::Statistics::DenseFrequencyContainer2 > HistogramType;

typedef itk::Statistics::SampleToHistogramFilter<
  ListSampleType, HistogramType > HistogramFilterType;

HistogramFilterType::Pointer histogramFilter = HistogramFilterType::New();
```

The use of `itk::Statistics::SampleToHistogramFilter` can be refered to itk documentations [1, 2] for help.

```
typedef HistogramFilterType::HistogramType  HistogramType;

histogramFilter->SetInput( adaptor->GetSample() );

histogramFilter->SetMarginalScale( 10.0 );

typedef HistogramFilterType::HistogramSizeType HistogramSizeType;

MeasurementVectorType value;

unsigned int numberOfScalarComponents =
  itk::Statistics::MeasurementVectorTraits::GetLength( value );

HistogramSizeType histogramSize( numberOfScalarComponents );
histogramSize[0] = atoi( argv[2] );

histogramFilter->SetHistogramSize( histogramSize );

typedef HistogramFilterType::HistogramMeasurementVectorType HistogramMeasurementVectorType;

HistogramMeasurementVectorType histogramBinMinimum( numberOfScalarComponents );
histogramBinMinimum.Fill(  atof(argv[3]) - 0.5 );

HistogramMeasurementVectorType histogramBinMaximum( numberOfScalarComponents );
histogramBinMaximum.Fill(  atof(argv[4]) - 0.5 );

histogramFilter->SetHistogramBinMinimum( histogramBinMinimum );
histogramFilter->SetHistogramBinMaximum( histogramBinMaximum );

histogramFilter->Update();
```

The result of `itk::Statistics::SampleToHistogramFilter` can be printed out on the screen.

```
HistogramType::ConstPointer histogram = histogramFilter->GetOutput();

const unsigned int histogramTotalNumberOfBins = histogram->Size();

std::cout << "Total Number of Bins " << histogramTotalNumberOfBins << std::endl;

for( unsigned int bin=0; bin < histogramTotalNumberOfBins; bin++ )
  {
  std::cout << "bin = " << bin << " frequency = ";
  std::cout << histogram->GetFrequency( bin, 0 ) <<std::endl;
  }
```

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

```
QuadEdgeMeshHistogram inputMesh.vtk \
NumberOfBins Min Max
```

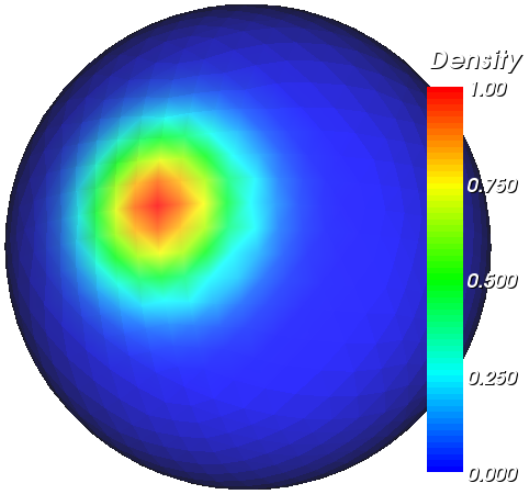

Figure 1: Radius = 100.0; Scalar values are between 0.0 and 1.0.

4 Results

Figure 1 shows the input mesh. The radius of it is 100.0 and it is centered to the origin of the coordinate system. The number of points on the mesh is 2,048. The scalar values of the points are between 0.0 and 1.0.

Figure 2 shows the histogram of the input mesh. X axis has the range of scalar values of the input mesh and Y axis shows the frequencies of each value.

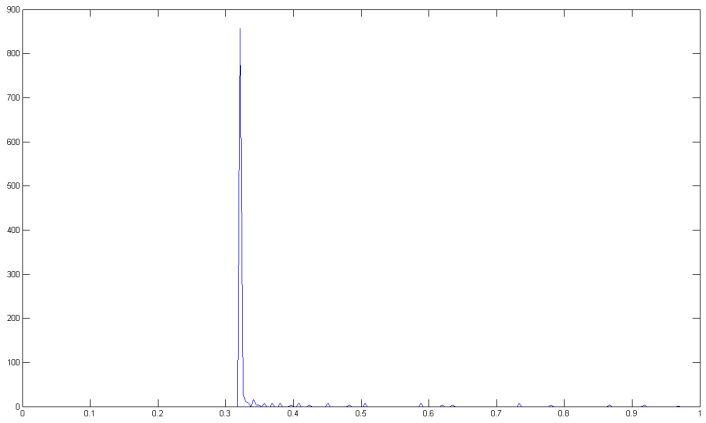

Figure 2: X axis: the range of scalar values; Y axis: the frequency of each value

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. (document), 3

[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. (document), 3