ITK Mesh IO Framework

Release 0.00

Wanlin Zhu[1]

September 8, 2010

[1]NeuroImaging Lab, NPI Institute, School of Psychiatry, University Of New South Wales, Australia

Abstract

ITK has a powerful and flexible image IO framework. Reading and writing different types of image file is straightforward. The image IO could be easily extended by writing a specific image IO class for a new image file format. Whereas, there is no such framework for easily reading and writing `itk::Mesh`/`itk::QuadEdgeMesh`, At the moment, only a few specific classes could read and write vtk polydata format and are not easily to be extended. This paper describes our contribution to itk for providing a mesh IO framework which could be used for reading and writing some commonly used mesh file formats. The mesh IO classes worked well for both `itk::Mesh` and `itk::QuadEdgeMesh`.

Contents

1 Introduction

ITK Provides a collection of classes for reading and writing a variety of image file formats. File type could be dynamically determined according to the information user input. Generally the file extension is used to determine which image IO object should be created for reading and writing the given image file. However, there are no corresponding classes for reading and writing `itk::Mesh`/ `itk::QuadEdgeMesh` which have been intensively used in the toolkit. We implemented the mesh IO framework for reading and writing mesh as simple as reading and writing image in itk.

2 Implementation

itk::MeshFileReader and itk::MeshFileWriter are two interface classes for reading and writing a variety of mesh file formats. itk::MeshIOBase is the base class for all mesh IO classes which perform reading and writing operation for a particular mesh format. itk::MeshIOFactory is derived from itk::ObjectFactoryBase, which could automatically create the MeshIOBase object according to given file extension. Currently we provide classes for reading and writing vtk legacy polydata data (*.vtk), Wavefront OBJ file(*.obj), Geomview Object File Format (*.off), BYU Geometry File Format (*.byu) and freesurfer ascii (*.fsa) and binary surface data (*.fsb). Since freesurfer surface file has no extension, we put *.fsa for ascii and *.fsb for binary mesh file respectively. Gifti mesh type is not supported in this release due to gifticlib is required to be included.

The interface between itk::MeshIOBase and itk::MeshFileReader/itk::MeshFileWriter doesn't take into account the point id or cell id information. which means current implementation could not handle data contains noncontinuous points, cells or point data or cell data. The reader always assume the input points, cells , point data and cell data are continuous.

The itk::MeshFileReader and itk::MeshFileWriter could read and write pixel type with itk::VariableLengthVector and itk::Array. In this case, the number of elements in pixel depends on the input data and determined until reading the input mesh. One of tests shows how it works.

We also provide functionality to read streamline mesh data. To do this, we define itk::PolylineCell which is modified from itk::PolygonCell.

3 Examples

The following example illustrates how to read itk::Mesh from input file and write it to the hard disk.

```
#include "itkMeshFileReader.h"
#include "itkMeshFileWriter.h"
#include "itkMesh.h"

  typedef float       MeshPixelType;
  const unsigned int Dimension = 3;

  typedef itk::Mesh< MeshPixelType, Dimension >   MeshType;
  typedef itk::MeshFileReader<MeshType>           ReaderType;
  typedef itk::MeshFileWriter<MeshType>           WriterType;

  ReaderType::Pointer reader = ReaderType::New();
  reader->SetFileName("input.vtk");
  try
  {
    reader->Update();
  }
  catch(itk::ExceptionObject & err)
  {
    std::cerr<<err<<std::endl;
    return EXIT_FAILURE;
  }

  //Do something with itk::Mesh

```

```
26  WriterType::Pointer writer = WriterType::New();
27  writer->SetFileName("output.vtk");
28  writer->SetInput(reader->GetOutput());
29  try
30  {
31    writer->Update();
32  }
33  catch(itk::ExceptionObject & err)
34  {
35    std::cerr<<err<<std::endl;
36    return EXIT_FAILURE;
37  }
```

In the `Examples` directory of the accompanying source code you will find an example to convert input mesh format to output mesh format.

4 Sofrware Requirements

Following software packages are required:

- CMake 2.4 or above
- Insight Toolkit 3.16 or above

References

[1] 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.