Mesh Similarity Calculator

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 itkQuadEdgeMeshSimilarityCalculator. It takes inputs of two meshes and gives the similarity of the labels on them in two types of similarity measurements – Dice Similarity and Jaccard Similarity.

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 deals with two meshes which have labels on them. Labels are represented as scalars of points on the meshes and the pixel types are required to be integer. It also requires two input meshes have the same topologies (number of points, number of cells, and number of edges).

Dice similarity, named after Lee Raymond Dice [1] and also known as the Dice coefficient, is a similarity measure related to the Jaccard index.

For sets X and Y of points which have the same label on different meshes, the coefficient may be defined as:

$$\frac{2\,|X \cap Y|}{|X| + |Y|} \tag{1}$$

Jaccard similarity, also known as the Jaccard index, is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard similarity measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$\frac{|X \bigcap Y|}{|X \bigcup Y|} \tag{2}$$

2 Method

This filter can give the value of both Dice and Jaccard similarities for a particular label on the meshes. The measurement for a label on the mesh is calculated by the sum of area which has the particular label.

Labels on the meshes are given as the scalar values of them, and the area of the label is calculated by the area of triangles which have the particular labeled points. If there is only one point in a triangle labeled by the particular value, a third of the triangle's area is added up into the sum of area belongs to that label. If there is two points, two third will be added.

So the filter goes through the cells and points on each cell one by one. Both of Dice and Jaccard similairties are calculated at the same time.

3 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

- itkQuadEdgeMeshSimilarityCalculator.h
- itkQuadEdgeMeshSimilarityCalculator.txx

The testing code is in Examples directory as

- QuadEdgeMeshSimilarityCalculator.cxx

3.1 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.

3.2 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/Report005`.

4 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.

This filter requires two input meshes have exactly the same number of points, number of cells and edges. And the meshes should have the same radius too.

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

- QuadEdgeMeshSimilarityCalculator

In order to use this filter we should include the header for the Mesh Similarity Calculator, the reader and the `itk::QuadEdgeMesh` itself.

```
#include "itkQuadEdgeMeshVTKPolyDataReader.h"
#include "itkQuadEdgeMeshSimilarityCalculator.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 int      MeshPixelType;
  const unsigned int Dimension = 3;

  typedef itk::QuadEdgeMesh< MeshPixelType, Dimension >   InputMeshType1;
  typedef itk::QuadEdgeMesh< MeshPixelType, Dimension >   InputMeshType2;
```

In order to read the input meshes we declare reader types for all of the input meshes and create one instance of each one.

```
  InputReaderType1::Pointer inputMeshReader1 = InputReaderType1::New();
  inputMeshReader1->SetFileName( argv[1] );

  InputReaderType2::Pointer inputMeshReader2 = InputReaderType2::New();
  inputMeshReader2->SetFileName( argv[2] );
```

Since this filter is not using `Update()` to execute, readers should be updated before being passed to it and it is always safer to do it in a `try/catch` block.

```
  try
    {
    inputMeshReader1->Update( );
    inputMeshReader2->Update( );
    }
  catch( itk::ExceptionObject & exp )
    {
    std::cerr << exp << std::endl;
    return EXIT_FAILURE;
    }
```

We declare the type of the mesh similarity calculator and instantiate it.

```
  typedef itk::QuadEdgeMeshSimilarityCalculator< InputMeshType1, InputMeshType2 > SimilarityCalcu

  SimilarityCalculatorType::Pointer similarityCalculator = SimilarityCalculatorType::New();
```

The output of the readers is passed as input to the mesh similarity calculator.

```
  similarityCalculator->SetInputMesh1 (inputMeshReader1->GetOutput());
  similarityCalculator->SetInputMesh2 (inputMeshReader2->GetOutput());
```

A particular label needs to be set to the filter by using `SetLabelValue()`.

```
similarityCalculator->SetLabelValue (atoi(argv[3]));
```

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

```
similarityCalculator->Compute();
```

We can get the value of Dice and Jaccard by calling `GetDice()` and `GetJaccard()`. We let them be printed on the screen in the `Example`.

```
//print out the values of similarities
std::cout<<"the Dice measure of Label"<< argv[3];
std::cout<<" is: "<<similarityCalculator->GetDice()<<std::endl;
std::cout<<"the Jaccard measure of Label"<< argv[3];
std::cout<<" is: "<<similarityCalculator->GetJaccard()<<std::endl;
```

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

```
MeshSimilarityCalculator inputMesh1.vtk inputMesh2.vtk LabelValue
```

References

[1] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945. 1