Transforming an itkPointSet

Release 0.00

David Doria

July 29, 2009

Rensselaer Polytechnic Institute, Troy NY

Abstract

This document presents a set of classes to enable operations on itkPointSet objects. In particular, itk-TransformPointSetFilter allows a transformation to be applied to a set of points. We propose these classes as addition to the Insight Toolkit ITK `www.itk.org`.

Contents

The Insight Toolkit provides a container, itkMesh, for data sets which contain geometry and topology. Another container, itkPointSet is provided for data sets consisting only of points (no topology). However, many functions which apply to both meshes and point sets are implemented only for meshes. Users have been required to instantiate a topology-less itkMesh in order to perform operations (for example, transformations) to their point set data. This set of classes removes that requirement and allows point sets to be operated on more intuitively.

1 Transforming an itkPointSet

As to mirror the existing itkTransformMeshFilter class structure, we created itkPointSetSource and itk-PointSetToPointSetFilter as supporting classes for itkPointSetTransformFilter. The PointSetTransformFilter is designed to work exactly as the existing MeshTransformFilter. The following code snipped demonstrates how to apply a transform to an itkPointSet:

```
// Declare the pointset pixel type.
 typedef itk::PointSet<double, 3 > PointSetType;
```

```
// Declare the type for PointsContainer
typedef PointSetType::PointsContainer     PointsContainerType;

// Declare the type for PointsContainerPointer
typedef PointSetType::PointsContainerPointer
                                      PointsContainerPointer;
// Declare the type for Points
typedef PointSetType::PointType PointType;

// Create an input PointSet
PointSetType::Pointer inputPointSet = PointSetType::New();

// Insert data in the PointSet
PointsContainerPointer  points = inputPointSet->GetPoints();

// Fill the PointSet with data

PointType p;
p[0] = 1.0; // x coordinate
p[1] = 2.0; // y coordinate
p[2] = 3.0; // z coordinate
points->InsertElement( count, p );

//... continue filling point set...

// Declare the transform type
typedef itk::AffineTransform<float,3> TransformType;

// Declare the type for the filter
typedef itk::TransformPointSetFilter<
                              PointSetType,
                              PointSetType,
                              TransformType  >       FilterType;

// Create a Filter
FilterType::Pointer filter = FilterType::New();

// Create an  Transform
TransformType::Pointer   affineTransform = TransformType::New();
TransformType::OffsetType::ValueType tInit[3] = {100,200,300};
TransformType::OffsetType   translation = tInit;
affineTransform->Translate( translation );

// Connect the inputs
filter->SetInput( inputPointSet );
filter->SetTransform( affineTransform );

// Execute the filter
filter->Update();
```