Class for Serial Transformations

Release 0.00

Torsten Rohlfing[1]

February 13, 2007

[1]Neuroscience Program, SRI International, Menlo Park, CA

Abstract

This paper describes a simple class for serial transformations, i.e., the concatenation of other transformations. This can be used for `itk::ResampleImageFilter`.

Contents

1 Design Goals

The purpose of this class is to provide a transparent way of concatenating multiple coordinate transformations for use in `itk::ResampleImageFilter`. As such, the class itself implements the necessary components of the interface `itk::Transform` class interface.

2 Example

First, let us declare the dimension of the coordinate space and the type of the component transformation.

```
const unsigned int SpaceDimension = 2;
typedef double CoordinateRepType;

typedef itk::CenteredSimilarity2DTransform< CoordinateRepType > SimilarityTransformType;
itk::TransformFactory<SimilarityTransformType>::RegisterTransform();
```

Next, declare the serial transformation. Note that only coordinate space dimension and scalar coordinate data type are template arguments to the serial transformation, but *not* the type of the component transformations (which is arbitrary).

```
typedef itk::SerialTransform<CoordinateRepType,ImageDimension> SerialTransformType;
typedef SerialTransformType::TransformType TransformType;
SerialTransformType::Pointer serialTransform = SerialTransformType::New();
```

Now we shall read all transformations given on the command line:

```
typedef itk::TransformFileReader TransformReaderType;
typedef TransformReaderType::TransformListType TransformListType;

for ( int i = argcTransformList; i < argc; ++i )
  {
  TransformReaderType::Pointer transformReader = TransformReaderType::New();
  transformReader->SetFileName( argv[i] );
  transformReader->Update();

  TransformReaderType::TransformListType* tlist = transformReader->GetTransformList();

  TransformReaderType::TransformListType::iterator it = tlist->begin();
  while ( it != tlist.end() )
    {
    serialTransform->AddTransform( (*it).GetPointer() );
    ++it;
    }
  }
```

And finally set the transformation in the resample filter, just like we would any "ordinary" transformation:

```
typedef itk::ResampleImageFilter<ReadImageType, WriteImageType> ResampleFilterType;
ResampleFilterType::Pointer resample = ResampleFilterType::New();
resample->SetTransform( serialTransform );
```

3 Limitations

The current implementation has the following limitations:

1. The class does not provide access to the parameter vectors of its components. Therefore, the serial transformation cannot be optimized in the ITK registration framework.
2. Input and output vectors of all component transformations must have the same dimension. It may of course be desirable to concatenate transformations for which this is not the case, e.g., an affine 3-D transformation followed by a 3-D to 2-D projection and a 2-D rigid transformation. This is not currently supported.
3. Input and output vectors of all component transformations must have the same scalar data type.

Acknowledgments

Work on this class was funded by the National Institutes on Alcohol Abuse and Alcoholism through the Integrative Neuroscience Initiative on Alcoholism (INIA) under Grant No. AA013521 (PI: A. Pfefferbaum).