An ITK Class that Splits Multi-Component Images

Release 1.0.0

Matthew McCormick[1]

July 15, 2012

[1]University of Wisconsin-Madison
Department of Biomedical Engineering
matt@mmmccormick.com

Abstract

This document describes the implementation of a class for the Insight Toolkit (ITK) that takes an image with multi-component pixels and outputs a scalar image for each component. This can be useful when examining images of vectors, tensors, etc. This filter may be easier to use and require fewer pixel visitations than an `itk::ImageAdaptor` with a similar purpose, but it requires additional memory allocation.

Contents

1 Introduction

Often when working with images that have pixels containing multiple components, it is desireable to address individual components. Further analysis may only be possible on separate components or one may simply want to view the components individually.

There are a few classes meant to address this problem in the Insight Toolkit (ITK). The `itk::NthElementAdaptor` and `itk::VectorImageToImageAdaptor` are `itk::ImageAdaptor`'s. They are useful when iterating over the image but only working with a single component.

Here we present a filter that produces multiple output images given a multi-component input image. This may be easier to work and may be more efficient than the adaptors when all the components of the image are desired to appear as scalar images. However, additional memory consumption is required compared to an adaptor. We call this class the `itk::SplitComponentsImageFilter`.

2 Class Overview

The `itk::SplitComponentsImageFilter` inherits from `itk::ImageToImageFilter`. The filter takes a multi-component image as an input, and multiple output scalar images are present on the output corresponding to the component number. The class extracts components of an `itk::Image` of `itk::Vector`'s, `itk::CovariantVector`'s, `itk::SymmetricSecondRankTensor`'s, or other classes that have the same interface. The pixel's interface must implement `ValueType operator[] ( unsigned int )`.

The class is templated over the input image type, the output image type, and the number of components in the input image pixels.

Implementation is multi-threaded. A subset of the outputs can be allocated and populated with the `SetComponentMask` method.

3 An Example

A sample executable that writes individual scalar component images from a multi-component input image is included with the source code. An excerpt follows:

```
typedef itk::SplitComponentsImageFilter< InputImageType, OutputImageType,
        TComponents > FilterType;
typename FilterType::Pointer filter = FilterType::New();
filter->SetInput( reader->GetOutput() );

typedef itk::ImageFileWriter< OutputImageType > WriterType;
typename WriterType::Pointer writer = WriterType::New();

std::ostringstream ostr;
for( unsigned int i = 0; i < TComponents; ++i )
  {
  ostr.str( "" );
  ostr << args.outputPrefix << "Component" << i << ".mha";
  writer->SetInput( filter->GetOutput( i ) );
  writer->SetFileName( ostr.str() );
  writer->Update();
  }
```

4 Acknowledgments

The authors would like to thank the NIH for their funding with grants T90DK070079 and R90DK071515.