A MultipleImageIterator for iterating over multiple images simultaneously

Release 1.00

Joël Schaerer[1]

February 19, 2014

[1]Bioclinica SAS, 60 avenue Rockefeller, 69008, Lyon, France

Abstract

This document describes an iterator designed to make iterating over multiple images more convenient

Contents

1 Introduction

Several applications such as multi-atlas segmentation require frequent iteration over multiple image volumes at the same time.

Doing so with the regular ITK iterators is tedious and error prone as it requires updating each iterator at end of each iteration. Failing to do so results in hard to debug errors and crashes.

The MultipleImageIterator is a simple wrapper class that tries to make this more convenient.

2 Design and implementation

The iterator is designed to share most of the itk image region iterator API.

The MultipleImageIterator holds a vector of image iterators. Most iterator methods are simply implemented by calling the corresponding method on all iterators. For example, the increment operator increments all iterators. The GoToBegin() method calls GoToBegin() on all iterators.

Defining the IsAtEnd() method is more challenging: checking all iterators would be correct, but very slow. The solution that was chosen is to perform full tests when the code is compiled in debug mode (NDEBUG is not set), and only check the first iterator in release mode. The user is expected to make sure all his iterators have the same number of values.

In addition to the usual iterator methods, the MultipleImageIterator provides the subscript operator ([]) which simply gives access to the underlying iterators.

One major limitation to the MultipleImageIterator is that all iterators must point to images of the same type, due to the lack of an untemplated base class for iterators in ITK.

3 Sample use

The following code demonstrates the use of the MultipleImageIterator in a program that draws random samples from a set of images and outputs them in CSV format:

```
int main()
{
  typedef itk::Image<float,3> ImageType;
  typedef itk::ImageFileReader<ImageType> ReaderType;

  typedef itk::ImageRegionIterator<ImageType> IteratorType;
  itk::MultipleImageIterator<IteratorType> it;

  std::string filenames[] =
{"originalT1.mha","csf_reg.nii.gz","grey_reg.nii.gz","white_reg.nii.gz"};
  std::vector<ImageType::Pointer> images; // Need to keep a reference as
iterators only have weak references
  ReaderType::Pointer r = ReaderType::New();
  for (unsigned int i=0; i<4;++i) {
    r->SetFileName(filenames[i]);
    r->Update();
    ImageType::Pointer im = r->GetOutput();
    im->DisconnectPipeline();
    images.push_back(im);
it.AddIterator(itk::ImageRegionIterator<ImageType>(im,im-
>GetLargestPossibleRegion()));
  }

  for (it.GoToBegin(); !it.IsAtEnd(); ++it) {
    if ((it[1].Get() != 0) && ((float)std::rand()) / RAND_MAX < 0.01) {
      for (unsigned int i=0; i<it.Size(); ++i) {
        std::cout << it[i].Get() << ";";
```

```
        }
        std::cout << std::endl;
      }
    }
}
```