MINC2.0 IO Support for ITK Revised

Leila Baghdadi
Mouse Imaging Centre,
Hospital for Sick Children,
Toronto, Ontario, Canada
baghdadi@sickkids.ca

MINC is a software system for storing and manipulating medical images, originally developed in 1993 by Peter Neelin at the McConnell Brain Imaging Centre. The name MINC is an acronym for Medical Imaging NetCDF. The file format was originally defined as a specialization of the NetCDF (Network Common Data Form) file format created by the Unidata Program Center at UCAR (University Corporation for Atmospheric Research). The NetCDF format, libraries, and tools were created to store generic datasets of arbitrary dimensionality. NetCDF was chosen because it implements many of the functions that were envisioned for the MINC system.

Introduction:

MINC2.0 has been designed to address a few specific problems that had been identified in MINC1.0 (original MINC).

- Limited file size. The NetCDF file format used 32-bit pointers to address objects within the file. This effectively restricted files to a maximum size of 2 gigabytes. With the advent of very high resolution brain atlas data (from macrotome or other sources) and large fMRI datasets, it became clear that this restriction might become a serious problem.
- Restricted data types. The NetCDF format defines a small fixed set of data types - integers, floating point, and ASCII strings. Neither aggregate data (arrays or structures) nor labeled (enumerated) data are supported as fundamental data types in NetCDF.
- Limited storage options. NetCDF files store data in a contiguous array. This inhibits the addition of either block addressable data or internal data compression to the NetCDF format.

Since most of these problems were inherent in the MINC1.0 file format, it was clear that the design of MINC2.0 would require a major revision of the file format. The team developing MINC2.0 chose to replace NetCDF with the HDF5 library to form the basis of the MINC2.0 format. HDF5 provides a number of advanced features which are not available in NetCDF.

This paper describes simple and convenient IO (read/write) access of MINC2.0 files from within ITK framework.

Description:

The pluggable factory pattern of ITK is used (mainly due to its simplicity of use) to register the "MINC2ImageIO" written specifically to place MINC2.0 function calls for reading and writing MINC2.0 Images.
This IO retrieves all the image information (dimension size, spacing, direction cosines and etc) and transfers them to ITK and further makes a separate call "miget_real_value_hyperslab" to retrieve the actual data (i.e., hyperslab: an N-Dimensional block of data) which is already scaled from within MINC2.0. For accuracy purposes, MINC2.0 uses the concept of slice scaling. This implies that each slice of the data has its own minimum and maximum values that must be taken into consideration when the voxel values are converted to real values.

NOTE: ITK has adopted the DICOM image format criteria in which the ORIGIN IS "NOT" rotated according to the direction cosines.
MINC2.0 format however requires the ORIGIN rotated based on the direction cosines. Therefore, MINC2 support for ITK makes sure that the origin passed to ITK is in fact "rotated" in advance.

The same procedure in reverse order is followed to write a MINC2.0 image. All the image information is retrieved from ITK and written in the image header using MINC2.0 function calls. The raw image buffer is then passed from ITK to one of the MINC2.0 hyperslab functions "miset_real_value_hyperslsab". To avoid having to allocate a huge chunk of memory (due to large image sizes), the raw data is actually written in slice by slice manner. Small chunks of data are copied into a temporary buffer one by one and send to the appropriate hyperslab function.

MINC2.0 is added to ITK as an "Advanced" option (default to off) mainly to ensure the smooth build of ITK regardless of the existence of the MINC2.0 libraries. Once this option is turned on, the location of NetCDF, HDF5 and MINC2.0 libraries and headers must be set.

NOTE: you "MUST" build BOTH "NetCDF and HDF5" before attempting to build MINC2.0 libraries.
Please refer to the wed sites provided below.

An example test file is provided for testing the MINC2.0 support for ITK from within ITK.

- Create a MINC2.0 file from scratch with some random information (Optional). You can check MINC2.0 tests files (part of MINC2.0 distribution) for further information.
- Read the MINC2.0 image using the MINC2.0 IO from ITK. A sample MINC2.0 image is provided which includes the three spatial dimensions and non-identity direction cosines.
- Print random image information (i.e., image origin, direction cosines, etc)
- Write MINC2.0 file using the MINC2.0 IO from ITK.

Additions to ITK:

- Insight/CMakeLists.txt
  - Add option USE_MINC2 to cmake

- Insight/CMake/FindMINC2.cmake
  - Add file FindMINC2 file to cmake

- Insight/Code/IO
  - itkMINC2ImageIO.cxx
  - itkMINC2ImageIO.h
  - itkMINC2ImageIOFactory.cxx
  - itkMINC2ImageIOFactory.h

- Insight/Testing/Code/IO/CMakeLists.txt
  - Change cmake to build this test

- Insight/Testing/Code/IO
  - itkMINC2ImageIOTest.cxx

Further Information:

MINC2.0: http://www.bic.mni.mcgill.ca/software/minc/
HDF5: http://hdf.ncsa.uiuc.edu/HDF5/
NetCDF: http://www.unidata.ucar.edu/software/netcdf/

Acknowledgement:

Robert D. Vincent, *Montreal Neurological Institute, Montreal, Canada.*
Dr David Gobbi, *Atamai Inc, London, Ontario, Canada.*
Dr Luis Ibanez, *Kitware Inc, New York, USA.*

MINC2.0 IO Support for ITK
Revised (July 6/2006)

Leila Baghdadi
Mouse Imaging Centre,
Hospital for Sick Children,
Toronto, Ontario, Canada
baghdadi@sickkids.ca

MINC is a software system for storing and manipulating medical images, originally developed in 1993 by Peter Neelin at the McConnell Brain Imaging Centre. The name MINC is an acronym for Medical Imaging NetCDF. The file format was originally defined as a specialization of the NetCDF (Network Common Data Form) file format created by the Unidata Program Center at UCAR (University Corporation for Atmospheric Research). The NetCDF format, libraries, and tools were created to store generic datasets of arbitrary dimensionality. NetCDF was chosen because it implements many of the functions that were envisioned for the MINC system.

Introduction:

MINC2.0 has been designed to address a few specific problems that had been identified in MINC1.0 (original MINC).

- Limited file size. The NetCDF file format used 32-bit pointers to address objects within the file. This effectively restricted files to a maximum size of 2 gigabytes. With the advent of very high resolution brain atlas data (from macrotome or other sources) and large fMRI datasets, it became clear that this restriction might become a serious problem.
- Restricted data types. The NetCDF format defines a small fixed set of data types - integers, floating point, and ASCII strings. Neither aggregate data (arrays or structures) nor labeled (enumerated) data are supported as fundamental data types in NetCDF.
- Limited storage options. NetCDF files store data in a contiguous array. This inhibits the addition of either block addressable data or internal data compression to the NetCDF format.

Since most of these problems were inherent in the MINC1.0 file format, it was clear that the design of MINC2.0 would require a major revision of the file format. The team developing MINC2.0 chose to replace NetCDF with the HDF5 library to form the basis of the MINC2.0 format. HDF5 provides a number of advanced features which are not available in NetCDF.

This paper describes simple and convenient IO (read/write) access of MINC2.0 files from within ITK framework.

Description:

The pluggable factory pattern of ITK is used (mainly due to its simplicity of use) to register the "MINC2ImageIO" written specifically to place MINC2.0 function calls for reading and writing MINC2.0 Images.
This IO retrieves all the image information (dimension size, spacing, direction cosines and etc) and transfers them to ITK and further makes a separate call "miget_real_value_hyperslab" to retrieve the actual data (i.e., hyperslab: an N-Dimensional block of data) which is already scaled from within MINC2.0. For accuracy purposes, MINC2.0 uses the concept of slice scaling. This implies that each slice of the data has its own minimum and maximum values that must be taken into consideration when the voxel values are converted to real values.

NOTE: ITK has adopted the DICOM image format criteria in which the ORIGIN IS "NOT" rotated according to the direction cosines.
MINC2.0 format however requires the ORIGIN rotated based on the direction cosines. Therefore, MINC2 support for ITK makes sure that the origin passed to ITK is in fact "rotated" in advance.

The same procedure in reverse order is followed to write a MINC2.0 image. All the image information is retrieved from ITK and written in the image header using MINC2.0 function calls. The raw image buffer is then passed from ITK to one of the MINC2.0 hyperslab functions "miset_real_value_hyperslsab". To avoid having to allocate a huge chunk of memory (due to large image sizes), the raw data is actually written in slice by slice manner. Small chunks of data are copied into a temporary buffer one by one and send to the appropriate hyperslab function.

MINC2.0 is added to ITK as an "Advanced" option (default to off) mainly to ensure the smooth build of ITK regardless of the existence of the MINC2.0 libraries. Once this option is turned on, the location of NetCDF, HDF5 and MINC2.0 libraries and headers must be set.

NOTE: you "MUST" build BOTH "NetCDF and HDF5" before attempting to build MINC2.0 libraries.
Please refer to the wed sites provided below.

An example test file is provided for testing the MINC2.0 support for ITK from within ITK.

- Create a MINC2.0 file from scratch with some random information (Optional). You can check MINC2.0 tests files (part of MINC2.0 distribution) for further information.
- Read the MINC2.0 image using the MINC2.0 IO from ITK. A sample MINC2.0 image is provided which includes the three spatial dimensions and non-identity direction cosines.
- Print random image information (i.e., image origin, direction cosines, etc)
- Write MINC2.0 file using the MINC2.0 IO from ITK.

Additions to ITK:

- Insight/CMakeLists.txt
  - Add option USE_MINC2 to cmake

- Insight/CMake/FindMINC2.cmake
  - Add file FindMINC2 file to cmake

- Insight/Code/IO
  - itkMINC2ImageIO.cxx
  - itkMINC2ImageIO.h
  - itkMINC2ImageIOFactory.cxx
  - itkMINC2ImageIOFactory.h

- Insight /Code/IO/CMakeLists.txt
  - Add options to link the corresponding libraries
    - netcdf, hdf5 and minc2

- Insight/Testing/Code/IO/CMakeLists.txt
  - Change cmake to build this test

- Insight/Testing/Code/IO
  - itkMINC2ImageIOTest.cxx

Further Information:

MINC2.0: http://www.bic.mni.mcgill.ca/software/minc/
HDF5: http://hdf.ncsa.uiuc.edu/HDF5/
NetCDF: http://www.unidata.ucar.edu/software/netcdf/

Acknowledgement:

Robert D. Vincent, *Montreal Neurological Institute, Montreal, Canada.*
Dr David Gobbi, *Atamai Inc, London, Ontario, Canada.*
Dr Luis Ibanez, *Kitware Inc, New York, USA.*