Enhancement Proposal

MINC2.0

(Draft)

December 20/ 2002

by

Leila Baghdadi and John G. Sled

Table of Contents

Abstract

The current release of the MINC libraries and software does not provide the functionality that is required for viewing and efficiently operating on the large data sets produced by the Mouse Imaging and the McConnell Brain Imaging Centres. This has led the interested parties to come up with a plan for the development of MINC2.0, which is described here.

Introduction

MINC (Medical Image NetCDF) is built on the general data format NetCDF. It consists of the low-level MINC library for interaction with NetCDF, a high-level C library (volume_io), which provides the volume data type, and another high_level C library (voxel_loop), for applying an operation to each voxel. Currently, the following features are supported by MINC

- An extensible header with a set of conventions for describing medical imaging data.
- Support for a variety of data types.
- Concept of a “world” coordinate system.
- Support for an arbitrary number of image dimensions.
- An API that abstracts many details of the data format and type.
- Multi-platform support.

More information on the above can be found at the following web sites:

- http://www.bic.mni.mcgill.ca/software/minc/
- http://www.bic.mni.mcgill.ca/software/minc/netcdf/guide.txn_toc.html
- http://www.bic.mni.mcgill.ca/~david/volume_io/volume_io.html

This document describes the proposal for MINC2.0 development and future implementation plans. The MINC2.0 project will be the result of a collaboration of the McConnell Brain Imaging Centre (BIC) at the Montreal Neurological Institute (MNI) and the MICe Imaging Centre in Toronto as well as other interested parties.

- MINC-at-MNI :
  - Bert Vincent
  - (to be filled later by people at MNI)

- MICe Imaging Centre:
  - John G. Sled (jgsled@sickkids.ca)
  - Leila Baghdadi (baghdadi@sickkids.ca)

- MINC Development:
  - Discussion and planning mailing list (min-development@bic.mni.mcgill.ca)

The remainder of this document contains a description of the requirements for the proposed software followed by a detailed outline of the project and finally the plan for implementing the changes, development and support.

Requirements

1) Large Datasets

- Efficient random access to large datasets.
- Access to files more than 2 gigabytes (GB) in size.
  The new implementation should support access to files larger than 2 GB on 32 bit machines. Note that the C library supports 64 bit file pointers on most recently released 32 bit operating systems.
- Processing of files larger than the available memory.
  While recent implementations of the C library support 64 bit file pointers on 32 bit architectures, it is not possible to allocate more than 2GB of memory on a 32 bit architecture. This is troublesome for applications that assume that a whole dataset can be kept in memory. Moreover most current desktop and laptop computers have less than 2 GB of memory, placing further restrictions.

  The new implementation should provide a uniform method of accessing files larger than 2 GB such that only a portion of the dataset is kept in memory on 32 bit machines while users of 64 bit machines can choose to work with the entire dataset in memory.

2) Multi-resolution

- A data format which allows efficient access to lower resolution versions of the data.

3) Compression

- Compression, which is optional, must be transparent to the user.
- The design should allow new compression formats to be easily incorporated.
- Support for compression formats that can take advantage of the multi-dimensional nature of the data.

4) Backward Compatibility

- MINC2.0 must be able to read/write the old file format.

Outline of Project

The following is a breakdown of the project outline into steps. These are ordered such that some of our more pressing goals can be achieved before the completion of the project.

1) Create a test suite for all the existing software

- An automated way of testing the functionality of the existing code should be developed. This should provide a means of verifying that the functionality of current applications is not altered as the MINC package evolves. Note that this testing should extend to the suite of MINC tools distributed with MINC.
- This can begin immediately and continue throughout the project.
- Note: Bert Vincent has started working on this.

2) Choose a documentation format and distribution method

A format and set of conventions should be selected for creating and distributing the documentation so that documentation can be written along with the code. Each piece of the software / library must be well documented by the developer so as to create a well-documented package which is easily accessible by the public.

3)Establish detailed requirements and produce a design

(Based on this document and the discussion on the list we will have a meeting in January to come up with the design)

- The proposed design will consist of the following three layers:
  - Low level access to the files on disk (NetCDF).
  - A library for working with MINC files on disk which abstracts the choice of low level library (libMINC).
  - A library for working with MINC files wholly or partly in memory (successor of volume_io/voxel_loop).

4)Define a new file structure

- The new library will use NetCDF with possible extensions to overcome file size limitations. The API must be abstract enough to allow for the substitution of another file format instead of NetCDF.
- The various file format options should be orthogonal so that any combination of the newly implemented features is permitted.

Blocking (2N-1 dimensions)

(Blocking refers to the partitioning of a given N-dimensional dataset into smaller size sub-blocks i.e., 16X16X16 or 8X8X8)

- The motivation for blocking is that nearby voxels in the dataset are also likely to be near one another in the file. Every voxel would have two addresses, one for the block and the other for voxels within the block. This scheme can be conveniently implemented by representing an N dimensional dataset as a 2N dimensional array. However, since the blocks in the fastest varying block dimension are adjacent to one another in the file, it is not necessary to group the blocks along this dimension. Hence, the blocks can be N-1 dimensional and the dataset can be represented by a 2N-1 dimensional array. Note that the presence of blocking will be made transparent by the API such individual voxels can be requested by the usual N dimensional index.

Multi-resolution

- MINC should support a multi-resolution representation of the data that would allow efficient access to various levels of reduced resolution datasets. This not only has obvious benefits for visualization tools, but may also be an asset in implementing segmentation and registration schemes that are multi-scale. One proposal for handling this is would be the“Harr wavelet decomposition”.
  - <u>Haar wavelet scheme</u> (Not determined whether will be implemented or not)

    Wavelets are a mathematical tool for representing functions hierarchically. The simplest example of a wavelet is the Haar wavelet which operates on data by calculating the sums and differences of adjacent elements. The Haar wavelet operates first on adjacent horizontal elements and then on adjacent vertical elements.

    Problem : modifying a single voxel in the decomposed image is expensive.

    Benefit : format is multi-resolution without increasing disk space usage.
  - <u>Multi-thumbnail scheme</u>

    (The file contains multiple versions of the data at different resolutions. For a three dimensional file this would increase the usage of the disk space by 1/7)

    Problem : This layout does not lend itself to the incremental transmission of additional image details over the network. In this respect, this technique compares unfavorably to the wavelet technique in which only a portion of an image needs to be retrieved in order to produce the next higher resolution version of the image.

    Benefit : Applications that modify the high-resolution data do not need to update all the lower resolution images until the final image is written out. The API should provide some functionality for flushing all the changes so that all the resolutions are updated.

Note: the desire for efficient read / write access to multi-resolution datasets appears to be incompatible with the performance benefits of wavelet decomposition for viewing files over a network. This issue requires further examination. Another aspect of multi-resolution that is unclear is how to handle higher dimensional datasets. For example, should multi-resolution only apply to spatial dimensions?

Compression

- Good compression appears to be incompatible with random access. Therefore, the proposal is to decompress the image data in its entirety when the file is opened and optionally compress it again when it is written to disk. The compression would be applied to the image data and not the header file. This will be implemented by uncompressing the whole dataset into a temporary disk space before being read. Also, the implementation should support both lossy and lossless compression techniques. It may be worth examining schemes that would allow partial uncompression of multiresolution datasets.
- Reference:
  - Ihm, I., and Park, S., "Wavelet-based 3D compression scheme for interactive visualization of very large volume data", *Computer Graphics Forum, Vol.18, No.1, pp.3-15, March 1999.* *http://www.ticam.utexas.edu/~hun/research.html*

Template format

- A template format (i.e. a MINC file with no image data) will be implemented for use in defining standard coordinate systems and samplings.

Label Format

- The header contains an entry to indicate that the voxel values are to be interpreted symbolically as labels.
- The header includes a dictionary describing each label by associating each voxel value with a string entry.
- The core MINC tools would be modified to treat labels differently from image data where appropriate. The implementation of the multi-resolution format would also take this into account.

Complex Numbers

- It has been proposed that complex numbers be included among the basic types but it is not clear whether there is sufficient demand for this functionality.

Slice Scaling

- It is not clear whether the current system of scaling every slice separately should be maintained for the new format.

5) Implementation of New Format Using Existing API

There are four aspects to updating the current libraries to support the new file format. Given the priority of functionality to work with large files (i.e., blocking), this functionality will be implemented first and multi-resolution and compression would be implemented in subsequent iterations.
The four aspects are :

- Re implement (libMINC) allowing the developers the flexibility to change the API. This would be allowed since most of the existing software relies on either voxel_loop or volume_io, whose APIs will still be supported.
- Re implement voxel_loop for the new libMINC preserving the existing API
- Re implement volume_io for the new libMINC preserving the existing API
- Update the MINC tools that are distributed with the MINC package

6) 64-bit architectures

MINC requires modification for 64-bit architectures. The new code should run on both 32 and 64 bit architectures.

7) New API, all the functionality of volume_io/voxel_loop

Name Space

- The new library should export a smaller number of functions and these should be given names that less easily clash with other libraries.

Default/simple API

- An additional set of functions should be provided to make it easy for programmers not familiar with MINC to implement common tasks.

- **Get_hyperslab**
  - An efficient method should be provided for getting and setting arbitrarily shaped N dimensional blocks from the dataset in a desired data type.

- **Data type issues**
  - The handling of numeric formats should be reexamined. In the present scheme when using volume_io there is one number type for the original file, another for the in memory volume and a third, either floating point or integer, that has meaning to the user. If this is combined with a caching mechanism then it would seem more sensible to cache this latter type rather than either of the former two. On a related note, perhaps it would also be helpful to distinguish between files that are meant to be interpreted as integers and those that are not.
  - In the current volume_io, the in memory representation can have an arbitrary data type which is then translated to a real number by the set/get operations. In the new scheme, the in-memory representation of the data will be the final data type that the user requests so that it could be used without translation.

- **Header**
  - The API will be extended to allow reading and manipulation of optional information from the header.

- **Caching**
  - The new library would implement caching of the in-memory representation in a way that took advantage of the disk representation of the data (i.e, block size/layout).

- **Label volume**
  - The API should be extended to deal with labels and label dictionaries.

- **Multi-resolution**
  - The API should be extended to allow the access to the different resolution images and generation of these images.

- **Compression**
  Reading of compressed files will be transparent. The API would allow the user to control the type of compression used when writing the file.

- **Large files**
    - the library would allow processing of files larger than core through the caching mechanism. This mechanism would also allow files larger than 2GB to be used on 32 bit architectures.
- **voxel_loop**
    - A set of functions for performing the same operation on every voxel in the dataset.
    - The new library should provide similar functionality thus eliminating the need to switch between the voxel_loop and volume_io methods of file access.

8) Portability

- The build system should be revised as needed to support compilation on both UNIX and WINDOWS operating systems.

9) Parallelization

- Will be implemented in response to specific needs.
- The voxel_loop functionality lends itself to simple parallelization.

10) MINC Distribution

- Once developed, the distribution of the MINC software/libraries must be supported (i.e., binary RPMS and tar-distribution of source files ready for compilation).

11) Implementation Languages

- MINC2.0 will be a set of new libraries written entirely in the C language.
- C++ and Python APIs will be developed which capture the functionality of the new high level library (i.e., the replacement for volume_io/voxel_loop).

Implementing the Changes

The implementation of MINC2.0 will be done through the collaboration of MINC-at-MNI group with the individuals at the MICe Imaging Centre (refer to Introduction for details). The implementation will start sometime in the first months of 2003 and changes are expected to be completed within one year.