Introduction to MINC 2.0

Tutorial #1

Leila Baghdadi

May 26/2004

Outline

- Brief Background
  - what about MINC tools
- The MINC2.0 API
  - open/close volumes
  - dimensions
  - data type/class
  - reading/writing data
  - voxel/hyperslab functions
  - volume properties
  - volume creation
  - examples
- Acknowledgements

Background

- A medical image file format based on hdf5
  - block-structured storage
  - labeled volumes
  - internal compression
  - multi-resolution images
- What about MINC Tools ?
  - input – Tools automatically “do the right thing”
  - output – User must explicitly specify “-2” flag for hdf5
    - mincreshape input.mnc output.mnc -start 40 -count 1 -2

MINC2.0 Files

- Convert MINC to MINC2.0
  - mincconvert -2 input.mnc output.mnc
- Create MINC2.0 files from scratch
  - no need to have a knowledge of hdf5
  - must write a simple C program
    - include (minc2.h) with library (libminc2.a)
  - must compile it while linked against MINC2.0
    - -lminc2 -lhdf5 –lnetcdf

Open/Close VOLUME

- Handle
  - key to access volume and all its information
    - (mihandle_t)
- where to find the file
  - (/home/camelot/...)
- Mode
  - controls read/write access
    - (MI2_OPEN_RDONLY, MI2_OPEN_RDWR)
- After using the file
  - close volume and release handle

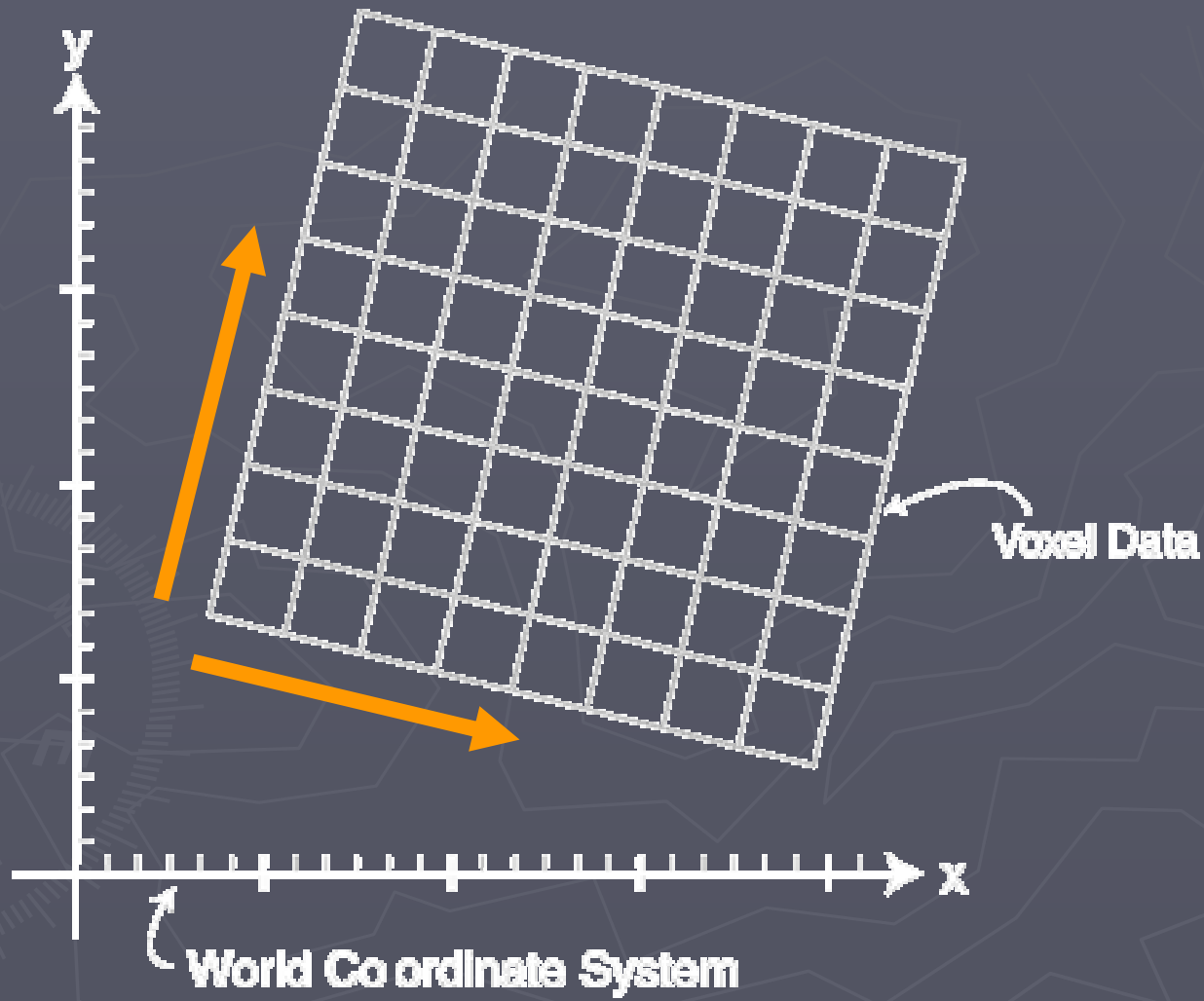
Example Code

```
#include <minc2.h>

main() {
int result;
mihandle_t hvol;

result =
    miopen_volume("/home/camelot/test.mnc",
    MI_OPEN_RDONLY, &hvol);

if (result == MI_ERROR) { /* MI_NOERROR */
    printf("Error opening the input file.\n");
}
/* work with the file here. */
miclose_volume(hvol);
}
```

Dimensions

- Handle
  - key to access all dimension information
    - (midimhandle_t)
- Name
  - standard names : xspace, yspace, zspace, time ..
- Class
  - MI_DIMCLASS_SPATIAL, MI_DIMCLASS_TIME, ..
- Attribute
  - regular sample (start, step)
  - irregular sample (start, offsets)
- Order
  - file, apparent
- Size
  - length
- Direction cosines
  - for Spatial dimensions only
- Units
  - unit of measurement (mm)
- Description
  - interpretation of the dimension

Direction Cosines



Create Dimension

- Only four essential properties must be specified
  - name, class, attribute, length
- Other properties will be set to default values
  - can be individually after dimension is created

micreate_dimension (name, class, attribute, length, dimension handle)

- Accessing or modifying any of the dimension properties is done by using

miget_dimension_”property”()

miset_dimension_”property”()

- Must free dimension if NOT associated with volume

mifree_dimension_handle (dimension handle)

Example Code

```
#include <minc2.h>
main() {
int result;
midimhandle_t hdim[3];
result =
micreate_dimension(MIxspace,MI_DIMCLASS_SPATIAL,
                   MI_DIMATTR_REGULARLY_SAMPLED,
                   10,&hdim[0]);
if (result == MI_ERROR) {
printf("Error creating dimension.\n");
}
micreate_dimension(MIyspace,MI_DIMCLASS_SPATIAL,
                   MI_DIMATTR_REGULARLY_SAMPLED,
                   20,&hdim[1]);
...
result = miset_dimension_description(hdim[0], “xspace increases from
                                                mouse left to right”);
result = miset_dimension_units(hdim[0],”cm”);
...
mifree_dimension_handle(hdim[0]);
}
```

Dimension Ordering

- Examples

file order (x,y,z) apparent order (z,y,x) → (z,y,x)	file order (x,y,z,t) apparent order (z,y,x) → (t,z,y,x)

- How to

1. set by dimension handle

   miset_apparent_dimension_order (
   volume handle, number of dimensions ,
   array of dimension handles)

2. set by dimension name

   miset_apparent_dimension_order_by_name (
   volume handle, number of dimensions ,
   array of dimension names)

Voxel Ordering

- Definition

file order (MI_FILE_ORDER)
direction is the SAME as the file order

counter file order (MI_COUNTER_FILE_ORDER)
direction is the OPPOSITE of file order

positive order (MI_POSITIVE)
voxel indices increase/decrease in direction of direction cosines if step is POSITIVE/NEGATIVE

negative order (MI_NEGATIVE)
voxel indices decrease/increase in direction of direction cosines if step is POSITIVE/NEGATIVE

- How to miset_dimension_apparent_voxel_order (dimension handle, order)

Other Dimension Functions

- micopy_dimension (
  dimension handle, new dimension handle)
- miget_volume_from_dimension (
  dimension handle, volume handle)
- miget_volume_dimensions (
  volume handle, dimension class, dimension attribute, dimension order, number of dimensions, array of dimension handles)
- miget_dimension_sizes (
  array of dimension handles, number of dimensions, array of dimension sizes)
- and MORE, check the API ...

Data Type/Class

- Format of the data stored in the file (TYPE)
  - MI_TYPE_BYTE,...,MI_TYPE_SHORT,..., MI_TYPE_SCOMPLEX (16 bit signed int), ..., MI_TYPE_UNKNOWN
- How the voxel data should be interpreted (CLASS)
  - MI_CLASS_REAL, MI_CLASS_INT, MI_CLASS_LABEL, MI_CLASS_COMPLEX, MI_CLASS_UNIFORM_RECORD, ...

Reading/Writing Data

- Functions operating on VOXELS
- Functions operating on HYPERSLABS
  - can be any size/dimensionality up to full image size
  - better (faster) to read/write data in large chunks

1. Functions operating on internal VOXEL value
2. Functions operating on REAL value

- The values in a minc file are ALWAYS interpreted as REAL, do not worry about internal representation

Real Value Data Functions

- VOXELS

miget_real_value
(volume handle, location, length, value pointer)

miset_real_value
(volume handle, location, length, value)

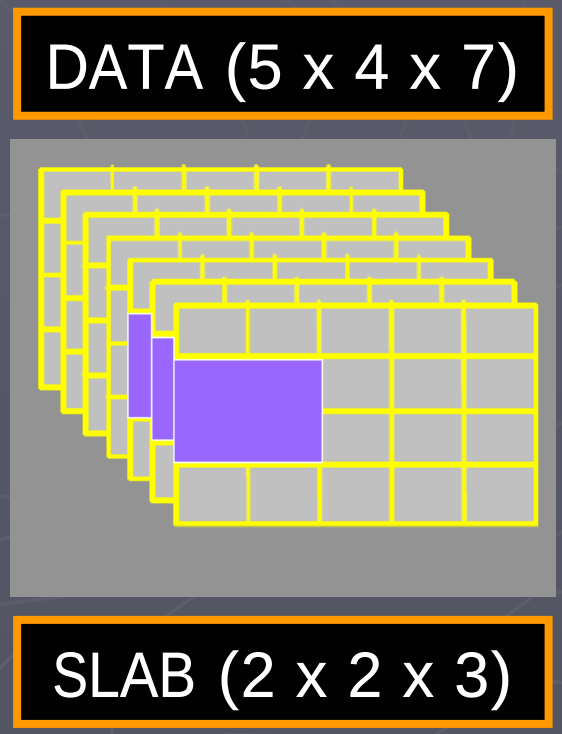
- HYPERSLABS

miget_real_value_hyperslab
(volume handle, type, start, count, buffer pointer)

miset_real_value_hyperslab
(volume handle, type, start, count, buffer pointer)

Hyperslabs

- need to specify arrays
  - start
  - count
    (size of hyperslab)
  - start + count < =
    Dimension length
  - apparent
    dimension & voxel order
    (default same as file order)



Hyperslabs Examples

file dimension length (X, Y, Z,T)

file order → x, y, z, t

apparent order → t, z, y, x

miget_voxel_value_hyperslab (..)

- Get entire file in one operation

  start[0]=start[1]=start[2]=start[3]=0

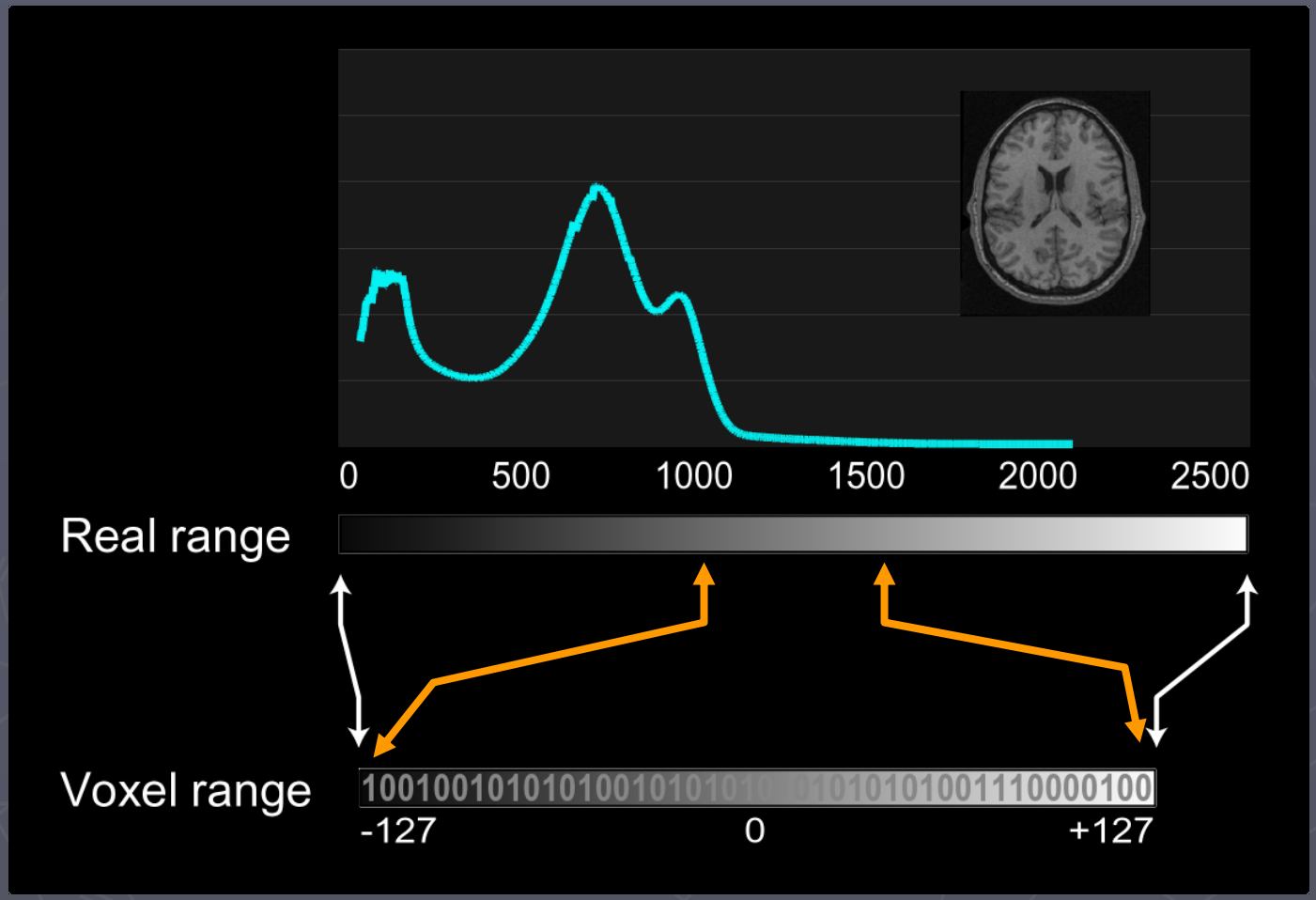
  count[0]= T; count[1]= Z; count[2]= Y; count[3]= X;

- Get data from one time location (T=1)

  start[0]=start[1]=start[2]=start[3]=0

  count[0]= 1; count[1]= Z; count[2]= Y; count[3]= X;

Intensity Mapping



- Global scale (whole volume)
  ONE map per VOLUME
- Slice
  ONE map per SLICE

Intensity Mapping (cont.)

- Voxel Range (min, max)
  - apply to volume as whole
  - range of the internal data type

  12 bit MRI (0..4095)

  16 bit unsigned (0..65535)

  16 bit signed (-32768..32767)
- Real Range (min, max)
  - defined for whole volume OR each slice independently
- Translation between voxel and real values

real value = voxel value * SCALE + SHIFT

SCALE = (real range) /(voxel range)

SHIFT = real min – (voxel min * SCALE)

Global versus Slice Scaling

- Global Scaling
  - default
    - scale_min = 0
    - Scale_max = 1
    - Has_slice_scaling = FALSE
  - set by user
    - volume_min,volume_max

      miset_volume_min/max (volume handle, volume_min/volume_max)

      OR
    - volume_range

      miset_volume_range (volume handle volume_min, volume_max)

- Slice Scaling
  - must set scaling flag →TRUE

    miset_slice_scaling_flag (volume handle, scaling_flag)
  - must set
    - slice_min, slice_max

      miset_slice_min/max (volume handle, start, length, slice_min/slice_max)

      OR
    - slice_range

      miset_slice_range (volume handle, start, length, slice_min, slice_max)

Volume Properties

- Must be defined before creating the volume
  - (mivolumeprops_t)
- Properties:
  - blocking
  - multi-resolution
  - compression
  - record size and field name
  - template volumes

Volume Properties Functions

- Create volume property

  minew_volume_props(volume property pointer)
- Destroy volume property

  mifree_volume_props(volume property)
- Set different properties

  miset_props_blocking(volume property, number of chunks, size of each chuck)

  miset_props_multi_resolution(volume property, enable flag, depth)

  miset_props_compression_type(volume property, compression type) (NONE, ZLIB)

  miset_props_zlib_compression(volume property, zlib_level)

  miget_volume_props(volume handle, volume property pointer)

Volume Creation

- Create file template (all information)

micreate_volume

(filename, number of dimensions, dimension handles, volume type, volume class, volume properties, volume handle pointer)

- Create actual image (must be called AFTER micreate_volume() is called)

micreate_volume_image

(volume handle)

Example

```
#include <minc2.h>
void check_for_error(int result);

main() {
int result;
mihandle_t hvol;
midimhandle_t dim[3];
mivolumeprops_t props;
unsigned long coords[3];

result = minew_volume_props(&props);
check_for_error(result);
result = miset_props_compression_type(props, MI_COMPRESS_ZLIB);
..
result = miset_props_zlib_compression(props, 3);
..
result = miset_props_multi_resolution(props, 1, 3);
..
```

Example (cont)

```
result = micreate_dimension("xspace",MI_DIMCLASS_SPATIAL,
    MI_DIMATTR_REGULARLY_SAMPLED,10,&dim[0]);
check_for_error(result);
result = micreate_dimension("yspace",MI_DIMCLASS_SPATIAL,
    MI_DIMATTR_REGULARLY_SAMPLED,10,&dim[1]);
...
result = micreate_dimension("zspace",MI_DIMCLASS_SPATIAL,
    MI_DIMATTR_REGULARLY_SAMPLED,10,&dim[2]);
...
result =
    micreate_volume("test_multi_h5.mnc", 3, dim,
    MI_TYPE_UINT, MI_CLASS_REAL,props,&hvol);
...
```

Example (cont)

```
result = micreate_volume_image(hvol);
...
for (i = 0; i < 10; i++) {
    for (j = 0; j < 10; j++) {
        for (k = 0; k < 10; k++) {
            coords[0] = i; coords[1] = j; coords[2] = k;
            result =
              miset_real_value(hvol, coords, 3, 1.0);
            check_for_error(result);
        }
    }
}}
void check_for_error( int result) {
    if (result == MI_ERROR) {
        printf(“ SOME ERROR OCCRUED \n”);
        return;
    }
}
```

Acknowledgements

- Bert Vincent
  - Design, documentation, and programming

  and LOTS and LOTS of helpful insights!
- John Sled
  - Design, documentation, and sanity checking
- Many others who have offered advice and opinions!
- NetCDF created by UCAR
- HDF5 created by NCSA