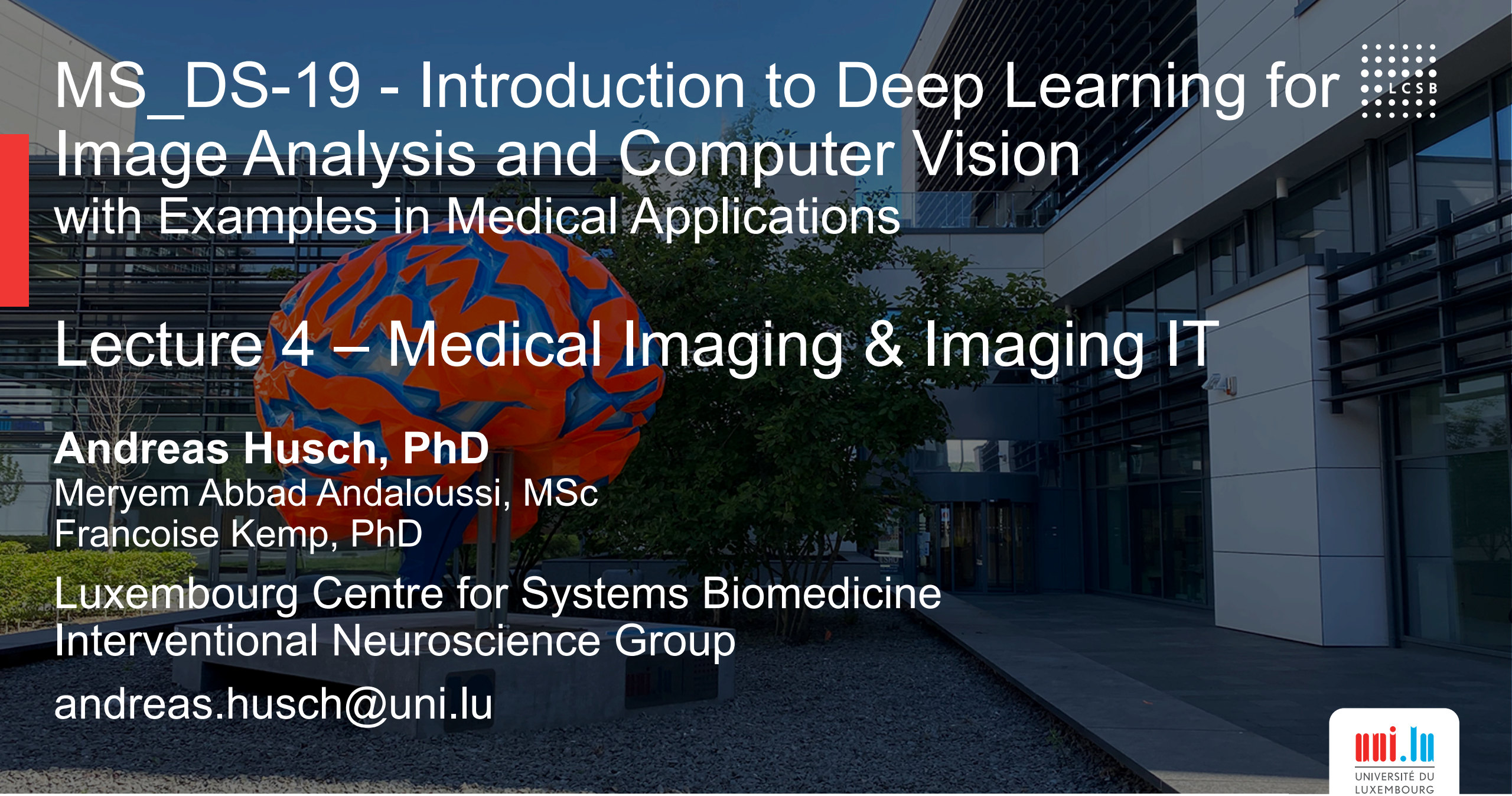


MS_DS-19 - Introduction to Deep Learning for Image Analysis and Computer Vision

with Examples in Medical Applications

Lecture 4 – Medical Imaging & Imaging IT

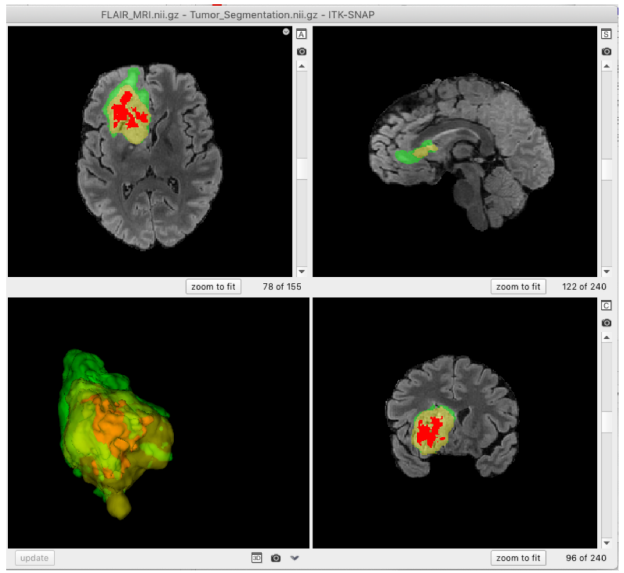
Andreas Husch, PhD
Meryem Abbad Andaloussi, MSc
Francoise Kemp, PhD

Luxembourg Centre for Systems Biomedicine
Interventional Neuroscience Group

andreas.husch@uni.lu

Recap last lecture

- Image Enhancement
- Filtering and Kernels
- Edge Detectors / Differential Operators
- Morphological Operators
- Segmentation



1	0	0	1
0	-1	-1	0

Δ_x Δ_y

Roberts Cross

1	1	1	1	0	-1
0	0	0	1	0	-1
-1	-1	-1	1	0	-1

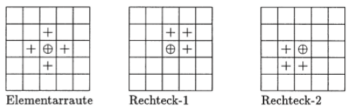
Δ_x Δ_y

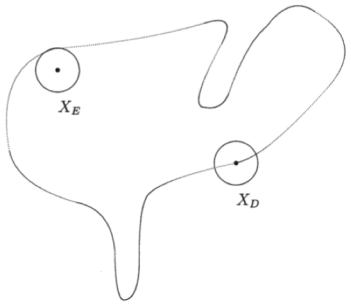
Prewitt

1	2	1	1	0	-1
0	0	0	2	0	-2
-1	-2	-1	1	0	-1

Δ_x Δ_y

Sobel





Imaging IT

Imaging IT - Typical General Purpose Raster Image Formats

- BMP
- JPEG
- TIFF
- PNG
- GIF
- …

Made for 2D imaging data, max. bit depth and potions for metadata varing by the format

Imaging IT - Typical General Purpose Raster Image Formats

Typical images properties

- BMP: 8 bit RGB (3 x 8 = *24 bit per pixel*) *uncompressed*
- JPEG: 8 bit RGB (3 x 8 = *24 bit per pixel*, with **lossy compression**)
- TIFF: 8 bit or 16 bit (!) RGB (3x8 or 3x*16=48 bit per pixel*), *lossless* compression possible, meta data in header possible, multiple image versions per file possible
- PNG: often 8 bit RGB**A** , *lossless* compressed, *4x8=32 bit per pixel*
- GIF: 256 colors using 8 bit colormap (*8 bit per pixel* + colormap in header)

Some formats allow the use with other bit depths, for example BMP also possible as 1 bit black and white or adding alpha channel

Imaging IT - Typical General Purpose Raster Image Formats

General Purpose Raster Image Formats are **not well suited** for medical imaging

- Limited range (for example 8 bit as uint8 i.e. 0..255
  - For example problems to encode a CT using Hounsfield units ranging from -1024 to 3072?
- Made for 2D data
  - Medical often 3D or even higher dimensional!
- No encoding of geometry
  - E.g. patient position in scanner; coordinate system of the image;
  - What is the “size” of a voxel in mm?
  - How do deal with anisotropic voxels eg 0.5mmx0x5mmx2mm?
- Encoding of metadata (patient name, sex, …) difficult

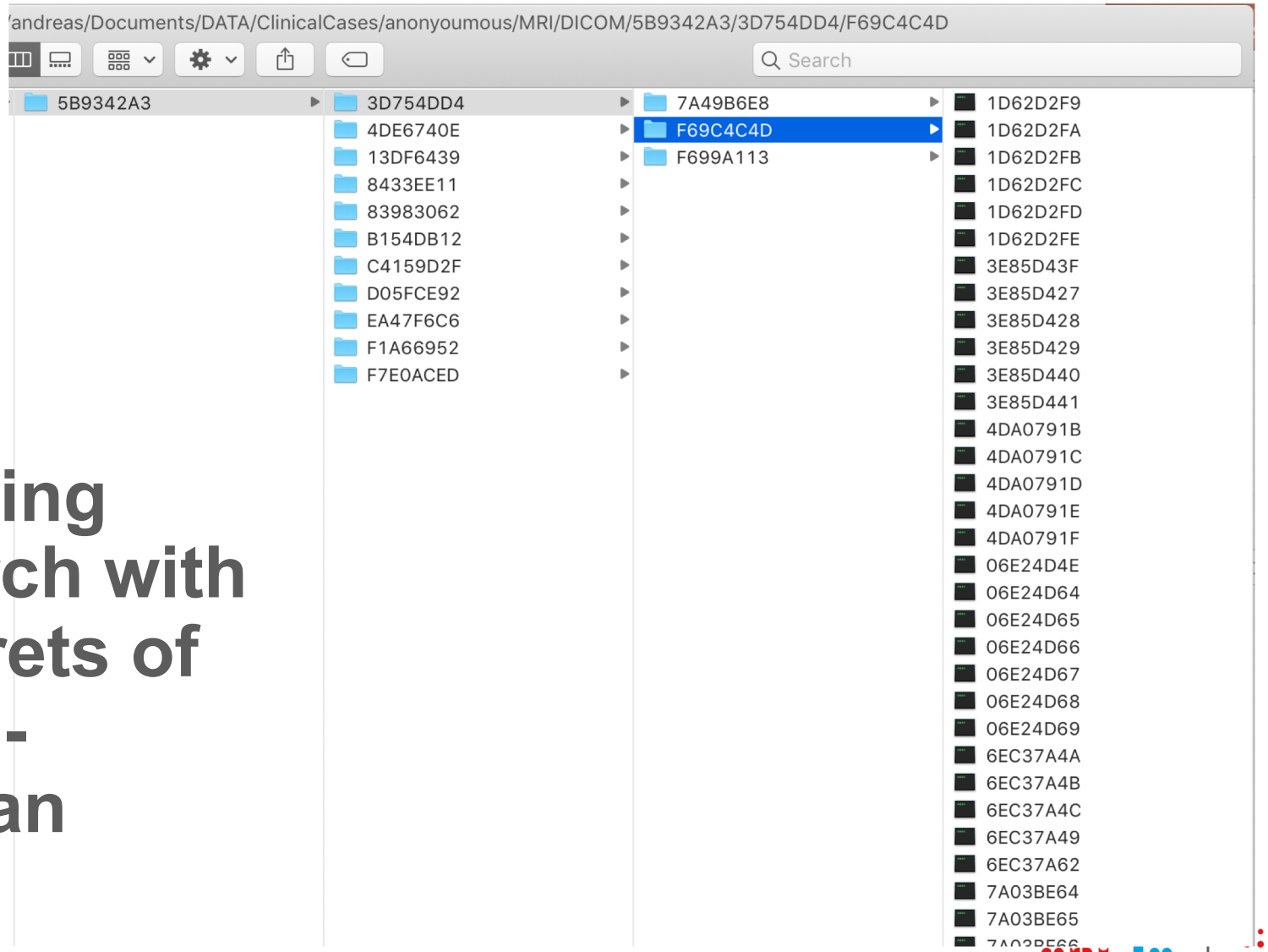
Imaging IT – Special Formats for Medicine

In ***Clinical*** **application: DICOM (.dcm)**

- Extremely **flexible format!**
- “Everbody” can define own **“tags”** an add specific metainformation to the -> enables product innovations! (e.g. a new scanner type that can encode things not know at the time the DICOM standard was defined**)**
- 3D imaging normally stored as “series” of 2D images, joined via header information (SeriesID tag)
- **Extremely annoying format for research with potentially hundrets of files for one high-resolution 3D scan**

Imaging IT – Special Formats for Medicine

- **DICOM**
  Extremely annoying format for research with potentially hundrets of files for one high-resolution 3D scan



Imaging IT – Special Formats for Medicine

In ***Research*** **: NiFTI (.niii / .nii.gz)**

- "One file per scan"
- Able to store *high dimensional data* (3D, 4D, …) in ***one*** file
  - More data processing (array) thinking
- Stores up to two transformation matrices in the header to encode image geometry ("VoxeI2World")
  - An Affine Matrix (homogenous coordinates)
  - A unit quaterion representation
- Does not store metadata – ensures a lot of data protection by design (can't accidently forget to remove the patient name)

Imaging IT – Special Formats for Medicine

In ***Research*: NiFTI (.nii / .nii.gz)**

- "One file per scan"
- Able to store *high dimensional data* (3D, 4D, …) in ***one* file**
  - More data processing (array) thinking
- Stores up to two transformation matrices in the header to encode image geometry ("Voxel two World")
  - An Affine Matrix (homogenous coordinates)
  - A unit quaterion representation
- Does not store metadata – ensures a lot of data protection by design (can't accidently forget to remove the patient name)

Imaging IT – Special Formats for Medicine

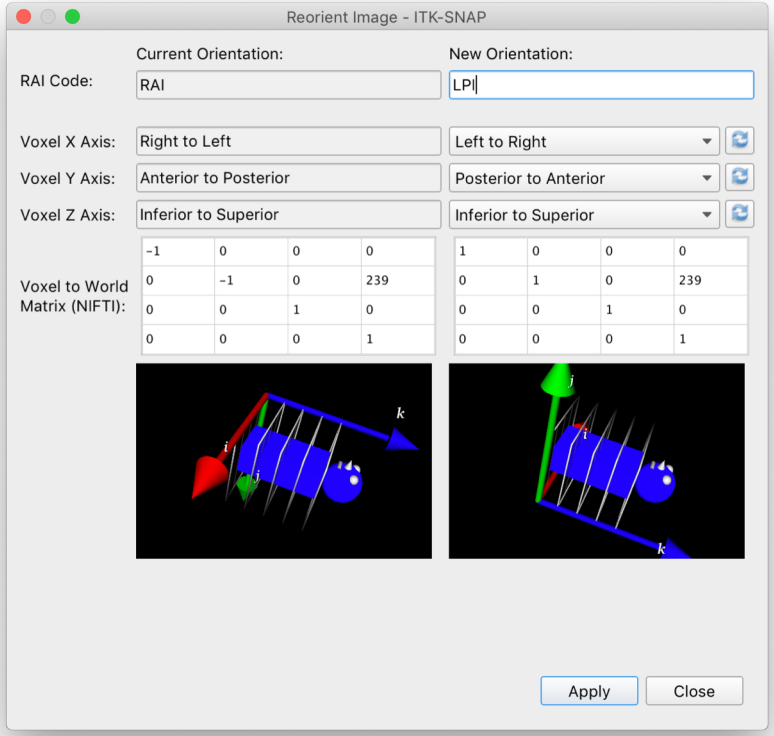
In *Research*: NiFTI (.nii / .nii.gz)

```
                    /**************************/ /**********************/ /************/
struct nifti_2_header {  /* NIFTI-2 usage           */ /* NIFTI-1 usage      */ /*  offset  */
                    /**************************/ /**********************/ /************/
   int   sizeof_hdr;     /*!< MUST be 540           */ /* int sizeof_hdr; (348) */   /*   0 */
   char  magic[8] ;      /*!< MUST be valid signature. */  /* char magic[4];    */   /*   4 */
   int16_t datatype;     /*!< Defines data type!    */ /* short datatype;       */   /*  12 */
   int16_t bitpix;       /*!< Number bits/voxel.    */ /* short bitpix;         */   /*  14 */
   int64_t dim[8];       /*!< Data array dimensions.*/ /* short dim[8];         */   /*  16 */
   double intent_p1 ;    /*!< 1st intent parameter. */ /* float intent_p1;      */   /*  80 */
   double intent_p2 ;    /*!< 2nd intent parameter. */ /* float intent_p2;      */   /*  88 */
   double intent_p3 ;    /*!< 3rd intent parameter. */ /* float intent_p3;      */   /*  96 */
   double pixdim[8];     /*!< Grid spacings.        */ /* float pixdim[8];      */   /* 104 */
   int64_t vox_offset;   /*!< Offset into .nii file */ /* float vox_offset;     */   /* 168 */
   double scl_slope ;    /*!< Data scaling: slope.  */ /* float scl_slope;      */   /* 176 */
   double scl_inter ;    /*!< Data scaling: offset. */ /* float scl_inter;      */   /* 184 */
   double cal_max;       /*!< Max display intensity */ /* float cal_max;        */   /* 192 */
   double cal_min;       /*!< Min display intensity */ /* float cal_min;        */   /* 200 */
   double slice_duration;/*!< Time for 1 slice.     */ /* float slice_duration; */   /* 208 */
   double toffset;       /*!< Time axis shift.      */ /* float toffset;        */   /* 216 */
   int64_t slice_start;  /*!< First slice index.    */ /* short slice_start;    */   /* 224 */
   int64_t slice_end;    /*!< Last slice index.     */ /* short slice_end;      */   /* 232 */
   char  descrip[80];    /*!< any text you like.    */ /* char descrip[80];     */   /* 240 */
   char  aux_file[24];   /*!< auxiliary filename.   */ /* char aux_file[24];    */   /* 320 */
   int qform_code ;      /*!< NIFTI_XFORM_* code.   */ /* short qform_code;     */   /* 344 */
   int sform_code ;      /*!< NIFTI_XFORM_* code.   */ /* short sform_code;     */   /* 348 */
   double quatern_b ;    /*!< Quaternion b param.   */ /* float quatern_b;      */   /* 352 */
   double quatern_c ;    /*!< Quaternion c param.   */ /* float quatern_c;      */   /* 360 */
   double quatern_d ;    /*!< Quaternion d param.   */ /* float quatern_d;      */   /* 368 */
   double qoffset_x ;    /*!< Quaternion x shift.   */ /* float qoffset_x;      */   /* 376 */
   double qoffset_y ;    /*!< Quaternion y shift.   */ /* float qoffset_y;      */   /* 384 */
   double qoffset_z ;    /*!< Quaternion z shift.   */ /* float qoffset_z;      */   /* 392 */
   double srow_x[4] ;    /*!< 1st row affine transform. */  /* float srow_x[4]; */   /* 400 */
   double srow_y[4] ;    /*!< 2nd row affine transform. */  /* float srow_y[4]; */   /* 432 */
   double srow_z[4] ;    /*!< 3rd row affine transform. */  /* float srow_z[4]; */   /* 464 */
   int slice_code ;      /*!< Slice timing order.   */ /* char slice_code;      */   /* 496 */
   int xyzt_units ;      /*!< Units of pixdim[1..4] */ /* char xyzt_units;      */   /* 500 */
   int intent_code ;     /*!< NIFTI_INTENT_* code.  */ /* short intent_code;    */   /* 504 */
   char intent_name[16]; /*!< 'name' or meaning of data. */ /* char intent_name[16]; */  /* 508 */
   char dim_info;        /*!< MRI slice ordering.   */      /* char dim_info;        */  /* 524 */
   char unused_str[15];  /*!< unused, filled with \0 */                                    /* 525 */
} ;                  /**** 540 bytes total ****/
typedef struct nifti_2_header nifti_2_header ;
```

Imaging IT – Image Orientation

There are **24 different** ways to order the image information in a 3D image.

- Unfortunately most of them have been used somewhere ;-)
- Often encoded using three letter code over the sets $\{R,L\}$, $\{I,S\}$, $\{A,P\}$

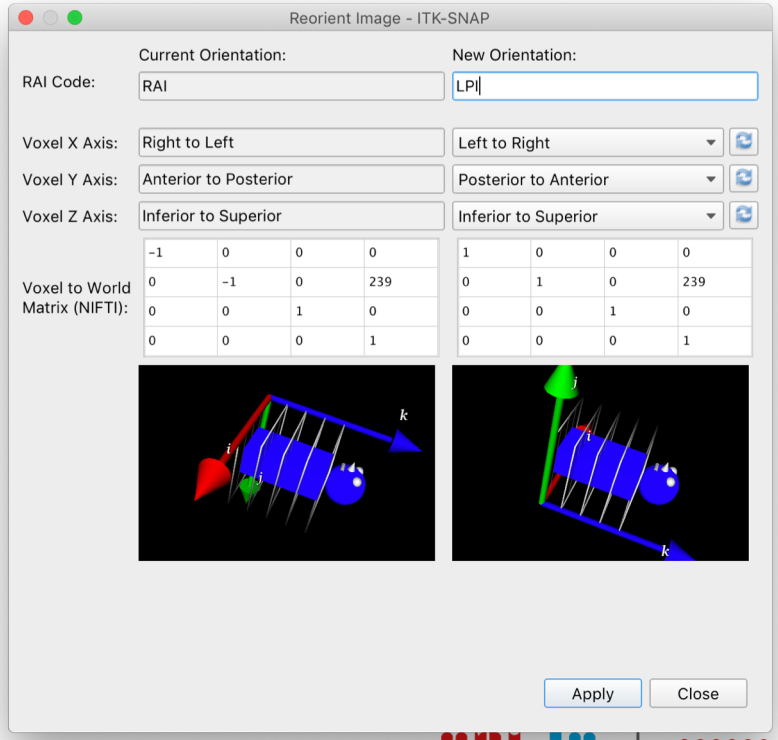


Imaging IT – **Image Orientation**

Most common:

- LPI, RAS and RAI
- Still not well defined as some people refer to “LPI” as
  - From L -> R, from P->A, from I -> S

  but others as
  - From R ->L, from A->P, from S -> I
- We will use a - to indicate the first form and + to indicate the second, i.e. LPI-.



Imaging IT – **Image Orientation**

Woxel2World Matrix

- Many libraries to load nifti data *apply* the `woxel2world` matrix stored in the nifty header *at loading time* to get a well defined image in mm scale.
- Ildeally you then always get a defined orientation in your software, regardless how the data was stored in the file, often *LPI-*
- Unfortunately the matrix might be wrong ☺
- Orientation mishaps might **not** be easily noticed. Why?