Orientation basics, Nifti 1.1 and Analyze 7.5 formats

(Maria del C. Valdés Hernández <M.Valdes-Hernan@ed.ac.uk>, 14.07.2014)

1.- Orientation basics

1.1.- DICOM

DICOM stands for Digital Imaging and Communications in Medicine, and it "is a standard for handling, storing, printing, and transmitting in medical imaging" (http://en.wikipedia.org/wiki/DICOM). Detailed information about this imaging standard can be found in its official website: http://dicom.nema.org/ .

Of extreme importance for image processing is to understand the image attributes that DICOMs provide. The Image Plane Module (in page 409 of part 3 of the standard) defines them:

Attribute name	Tag	Attribute description
Pixel Spacing	(0028,0030)	Physical distance in the patient between the centre of each pixel, specified by a numeric pair: adjacent row spacing (delimiter) adjacent column spacing in mm
Image Orientation (Patient)	(0020,0037)	The direction cosines of the first row and the first column with respect to the patient
Image Position (Patient)	(0020,0032)	The x, y and z coordinates of the upper left hand corner (centre of the first voxel transmitted) of the image in mm
Slice Thickness	(0018,0050)	Slice thickness in mm
Slice Location	(0020,1041)	Relative position of the image plane expressed in mm

Focusing in the orientation, DICOM defines a term: "Reference Coordinates System" or RCS where the X direction is from the patient's right hand side to his/her left's, the Y direction is from the front (i.e. forehead) to the back and the Z direction is from the patient's feet to the patient's head:

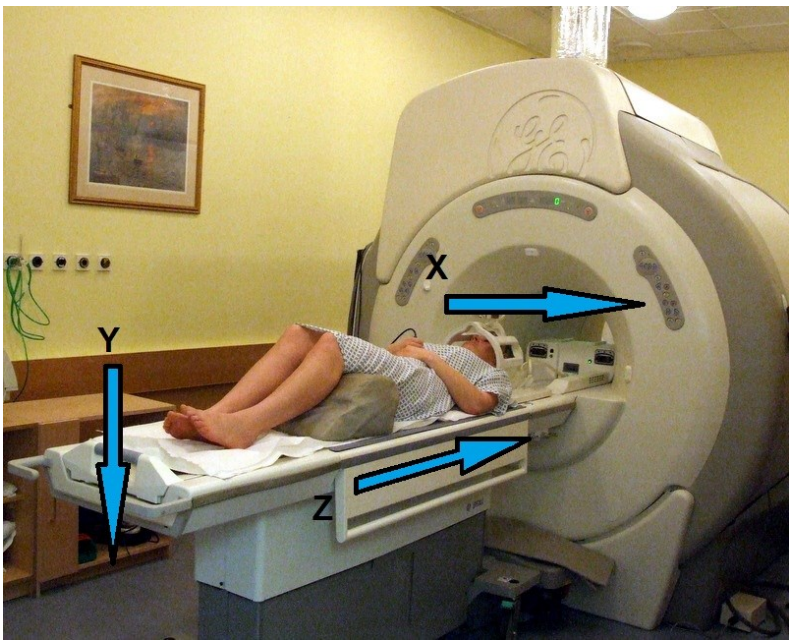


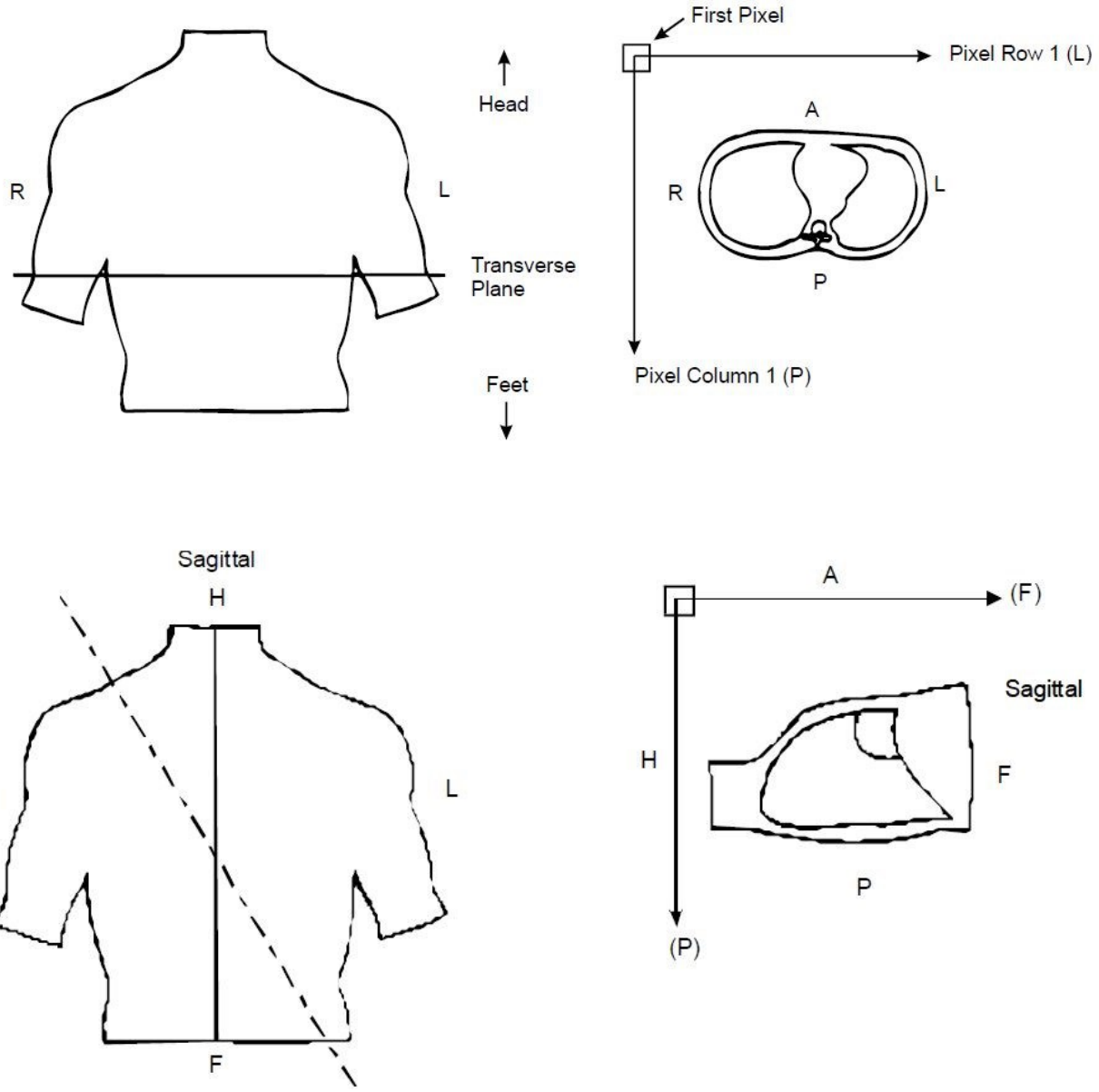
Figure 1. Reference Coordinate System for the scanner

The **X** direction increases from **Right to Left** (of the patient), the **Y** direction is (i.e. increases) from **Anterior to Posterior** (of the patient), and the **Z** direction is (i.e. increases) from **Inferior to Superior** (of the patient).

As a note of caution:

In http://freesurfer.net/fswiki/DICOM (accessed on 02.04.2014) it writes: "the DICOM coordinate system is the LPS (Left-Posterior-Superior)", which refers NOT to the origin, but to the OPPOSITE POINT, i.e. the one towards which the increase is made, according to what is defined in http://medical.nema.org/dicom/2003/03_03PU.PDF and is represented in the following screen-captures from the Annex E Explanation of patient orientation (Normative) PS3.3 -2003, page 711-

Note: Figure 2 below contains captures from the DICOM normative where H means Head (i.e. Superior) and F means Feet (i.e. Inferior)



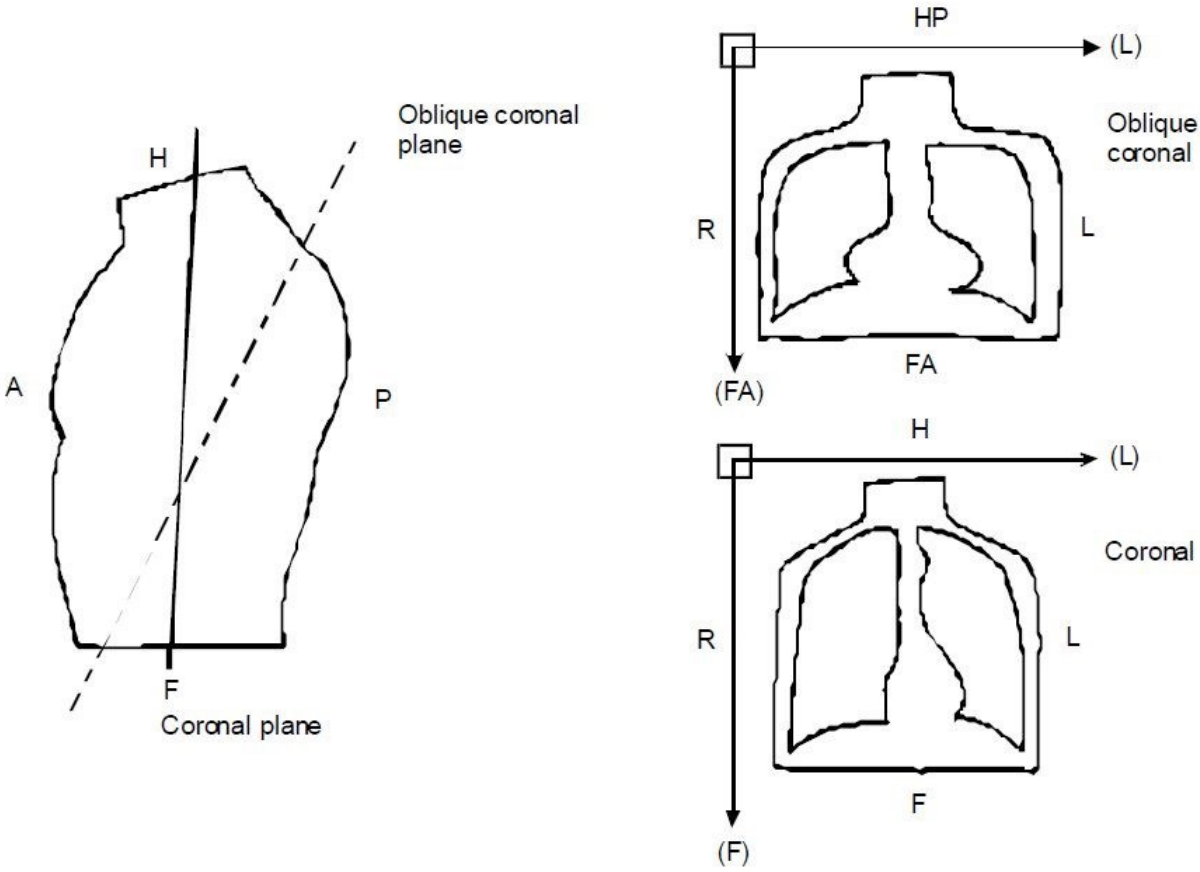


Figure 2. Snapshots of the Annex E of the DICOM standard

This is clarified in http://www.slicer.org/slicerWiki/index.php/Coordinate_systems , which adds that the LPS system (Right to **L**eft, Anterior to **P**osterior, Inferior to **S**uperior) is used in DICOMs and by the ITK toolkit, while the RAS system (Left to **R**ight, Posterior to **A**nterior, Inferior to **S**uperior) is used by 3D Slicer.

An important consideration is given to the terms: "radiological convention or radiological orientation" and "neurological convention or neurological orientation" in relation to the way images are viewed (i.e. displayed). As Figure 3 shows, the radiologist's right hand side corresponds to the patients' left hand side, while it is the opposite for the neurologist.

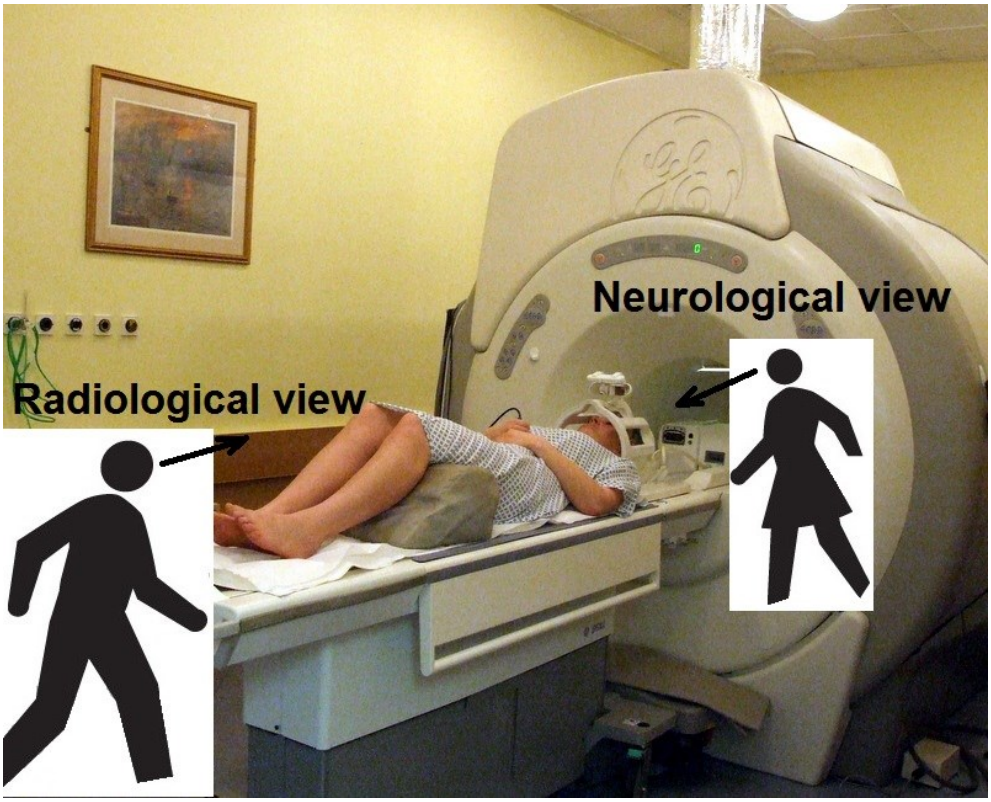


Figure 3. Radiological vs. neurological views

1.2.- Analyze 7.5 format

The Analyze 7.5 format was developed by the Biomedical Imaging Resource at Mayo Clinic to display and manipulate medical images. One data item consists of two files: one with the actual voxel data in a binary format (with the filename extension .img) and other named the header file with the information about how to manipulate the binary image data (i.e. contains voxel size, number of voxels on each dimension, etc.). **The header has 348 bytes, but when it is stored in the computer occupies 1Kbyte.** Detailed information can be found in http://eeg.sourceforge.net/ANALYZE75.pdf .

1.2.1.- MATLAB resources to manipulate Analyze7.5 format

The Image Processing Toolbox of MATLAB 12.0 and higher versions has incorporated 2 functions to manipulate Analyze 7.5 formatted files: **analyze75info** and **analyze75read**.

analyze75info: returns the header information on a 'struct'

analyze75read: returns the image data in neurological orientation: RAS (Left to **R**ight, Posterior to **A**nterior, Inferior to **S**uperior), **which is NOT the default orientation of Analyze 7.5 format**. Note that this MATLAB function reads 3D and 4D data represented as (y,x,z,t). Therefore, for working with a data array in the default Analyze 7.5 orientation, it is necessary to flip left and right and perform an in-plane rotation 90 degrees counterclockwise.

A useful library for manipulating both nifti 1.1(explained below) and Analyze 7.5 files is NIfTI, by Jimmy Shen, that can be downloaded from http://www.mathworks.co.uk/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image (website accessed on 02.04.2014).

1.3.- Nifti-1.1 format

Nifti-1.1 is a data format that resulted from an adaptation of the Analyze 7.5 format. Detailed information can be found at: http://brainder.org/2012/09/23/the-nifti-file-format/. Among its advantages over the original Analyze 7.5 format, the following can be mentioned:

- Affine coordinate definitions relating voxel index (I,j,k) to spatial location (x,y,z)
- Codes to indicate spatio-temporal slice ordering for fMRI
- Complete set of 8-128 bit + 256 data types
- Header and image files integrated in only one with extension .nii

The following table shows a comparison between Analyze 7.5 and nifti 1.1 headers and refers to hdrinfo as the field in the struct "header information". Descriptions follow MATLAB programming convention style.

Type	Field size	Offset Size (ana/ nifti)	Field name Analyze 7.5	Field name nifti 1.1	Description Analyze 7.5	Description nifti 1.1
struct header_key						
int32	4B	0B	sizeof_hdr		Size of the header. Must be 348 (bytes)	
char	10B	4B	data_type[10]		['dsr ' 0]	not used
char	18B	14B	db_name[18]		[filename(1:17) 0] hdrinfo.DatabaseName	not used
int32	4B	32B	extents		16384 hdrinfo.Extents	not used
int16	2B	36B	session_error		hdrinfo.SessionError	not used
char	1B	38B	regular		'r' hdrinfo.Regular must be 'r' to indicate all images and volumes are the same size	not used
char	1B	39B	dim_info		'0' not used	encodes directions (phase,frequency, slice). Possible values are 1, 2 or 3 for each. Spiral sequences have frequency and phase both encoded as 0.
struct image_dimensions → fseek(fid,40,'bof');						
int16	16B	40B	dim[8]		dim = [4 x y z t 1 1 1]; or dim = [3 x y z 1 1 1 1]; hdrinfo.Dimensions = size(data) = [x,y,z,t] or [x,y,z,1]	
char	2B	56B/ 68B	voxel_ units	intent_ code	'mm' hdrinfo.VoxelUnits	NIFTI intent code (complemented by the 3 fields below)
float	4B	58B/ 56B		intent_p 1	fwrite (fid,zeros(1,12), 'char') not used	The intent fields are codes to indicate the nature of the data, distribution, etc.
float	4B	62B/ 60B		intent_p 2		

float	4B	66B/ 64B		intent_p 3		See: http://brainder.org/2012/09/23/ the-nifti-file-format/ Posted on 23.September.2012 by A. M. Winkler
int16	2B	70B	TYPE	datatype	switch hdrinfo.ImgDataType case 'DT_NONE' TYPE = 0; case 'DT_UNKNOWN' TYPE = 0; case 'DT_BINARY' TYPE = 1; case 'DT_UNSIGNED_CHAR' TYPE = 2; case 'DT_SIGNED_SHORT' TYPE = 4; case 'DT_SIGNED_INT' TYPE = 8; case 'DT_FLOAT' TYPE = 16; case 'DT_COMPLEX' TYPE = 32; case 'DT_DOUBLE' TYPE = 64; case 'DT_RGB' TYPE = 128; case 'DT_ALL' TYPE = 255; end	UNKNOWN = 0 BINARY (1 bit) = 1 UNSIGNED_CHAR (1 byte) = 2 SIGNED_SHORT (2 bytes) = 4 SIGNED_INT (4 bytes) = 8 FLOAT (4 bytes) = 16 COMPLEX (8 bytes) = 32 DOUBLE (8 bytes) = 64 RGB (3 bytes) = 128 ALL = 255 signed char (1 byte) = 256 unsigned short (2 bytes) = 512 unsigned int (4 bytes) = 768 long long (8 bytes) = 1024 unsigned long long (8 bytes) = 1280 long double (16 bytes) = 1536 double pair (16 bytes) = 1792 long double pair (62 bytes) = 2048 RGBA (4 bytes) = 2304
int16	2B	72B	bitpix		hdrinfo.BitDepth (number of bits per voxel)	
int16	2B	74B	slice_start		0 (first slice index. Normally it is zero)	

float	32B	76B	pixdim[8]		v1 = hdrinfo.PixelDimensions(1);
					v2 = hdrinfo.PixelDimensions(2);
					v3 = hdrinfo.PixelDimensions(3);
					v4 = hdrinfo.PixelDimensions(4);
					pixdim = [v1 v2 v3 v4 0 0 0 0]; % Field for grid spacing (unit per dimension).	

float	4B	108B	vox_offset		hdrinfo.VoxelOffset (offset into the file)	
float	4B	112B		scl_slope	not used	Data scaling, slope
float	4B	116B		scl_inter	not used	Data scaling, offset
int16	2B	120B		slice_end	fwrite(fid,0,'float'); % not used	Last slice index
char	1B	122B		slice_ code		Slice timing order
char	1B	123B		xyzt_ units		Units of pixdim (1 to 4)
float	4B	124B	cal_max		hdrinfo.CalibrationMax = hdrinfo.GlobalMax-(hdrinfo.GlobalMax/100); % glmax saturated at 1% (maximum display intensity)	
float	4B	128B	cal_min		hdrinfo.CalibrationMin = 0; % minimum display intensity	
int32 /float	4B	132B	compres sed	slice_dur ation	0 (hdrinfo.Compressed)	Time for one slice
int32 /float	4B	136B	verified	toffset	0 (hdrinfo.Verified)	Time axis shift
int32	4B	140B	glmax		hdrinfo.GlobalMax = max(nonzeros(data));	Not used
int32	4B	144B	glmin		hdrinfo.GlobalMin =	Not used

					min(nonzeros(data));	
struct data_history → fseek(fid,148,'bof');						
char	80B	148B	descrip[80]		hdrinfo.Descriptor % any text up to 80 characters, for example: 'libBRIC file'	

char	24B	228B	aux_file[24]		if isequal(hdrinfo.AuxFile,'')
					aux_file = ['none ' 0]; % must be 24 bytes
					else
					aux_file = hdrinfo.AuxFile;
					end	

char/ short	1B/ 2B	252B	orient	qform	switch hdrinfo.Orientation
					case 'Transverse unflipped'
					orient = 0;
					case 'Coronal unflipped'
					orient = 1;
					case 'Sagittal unflipped'
					orient = 2;
					case 'Transverse flipped'
					orient = 3;
					case 'Coronal flipped'
					orient = 4;
					case 'Sagittal flipped'
					orient = 5;
					case 'Orientation unavailable'
					orient = 6;
					otherwise
					orient = 6; % assumed orientation unavailable
					end
						If 0: arbitrary coordinates compatible with Analyze
						If 1: Scanner-based anatomical coordinates = (sqrt(1- quatern_b^2- quatern_c^2- quatern_d^2), quatern_b, quatern_c, quatern_d)
						If 2: Coordinates aligned to another file (fields srow_*[4] below but referred to the scanner world's coordinates or to other image of the same subject)

uint16 /short	10B /2B	253B/ 254B	originato r[5]	sform	originator = [0 0 0 0 0];	Values from 0 to 4. If different from 0, relies on fields srow_*[4] below but referred to the standard space
char/ float	85B /4B	263B/ 256B		quatern_b	fwrite (fid,zeros(1,81), 'char'); % These are the fields for patient and scan details, which we fill with 0 for anonymising the scan	quatern_b parameter (see qform)
/float	/4B	/260B		quatern_c		quatern_c parameter (see qform)
/float	/4B	/264B		quatern_d		quatern_d parameter (see qform)
/float	/4B	/268B		qoffset_x		Quatern x shift
/float	/4B	/272B		qoffset_y		Quatern y shift
/float	/4B	/276B		qoffset_z		Quatern z shift
/float	/16B	/280B		srow_x [4]		1st row affine transform
/float	/16B	/296B		srow_y [4]		2nd row affine transform
/float	/16B	/312B		srow_z [4]		3rd row affine transform
/char	/16B	/328B		intent_n ame[16]		Name or meaning of the data (string of 16 characters)
char	4B	344B	magic[4]		[0 0 0 0] or ['ni1' 0] or [6EH 69H 31H 00H]	['n+1' 0] or [6EH 2BH 31H 00H]
		348B				