Geometry in Medical Imaging: DICOM and NIfTI formats

David Atkinson [orcid.org/0000-0003-1124-6666]

April 2022

1 Introduction

The spatial relation between the pixels in a medical image and their real-world position is important for clinical image display, surgery planning, image fusion and comparison of images acquired with different pixel sizes, orientations, scanners or time points.

The correct manipulation of this 'geometry' or 'spatial referencing' can be challenging. This paper aims to provide a clear description of the link between the data in the computer, array indexing and the 3D location where the image data was acquired.

Coordinate systems link coordinates to spatial locations. A scanner coordinate system is usually fixed relative to the room, whereas a patient coordinate system is fixed relative to the patient. The DICOM standard [1] uses an 'LPH+' patient coordinate system whereby the positive direction of the first dimension increases towards the patient Left, the second dimension increases towards the Posterior and the third dimension is towards the Head (or Superior). Note this LPH+ system relates to the patient and gives anatomical images with the same orientation, irrespective of whether the patient is head or feet first, prone or supine. The LPH system is right-handed. NIfTI [2] uses a RAH+ (Right-Anterior-Head) system, which is also right-handed.

Medical image data can be thought of as stored as a 1D line in memory or transmitted serially through a network. This data is accessed by using an array and subsequently displayed on a 2D screen. The ways an array can be indexed and mapped onto the screen can vary between systems and is a source of confusion. Nevertheless, the DICOM and NIfTI standards present unambiguous ways to assign a 3D position to any image data point. Figure 1 illustrates these concepts and the steps will be outlined in more detail for the DICOM and NIfTI

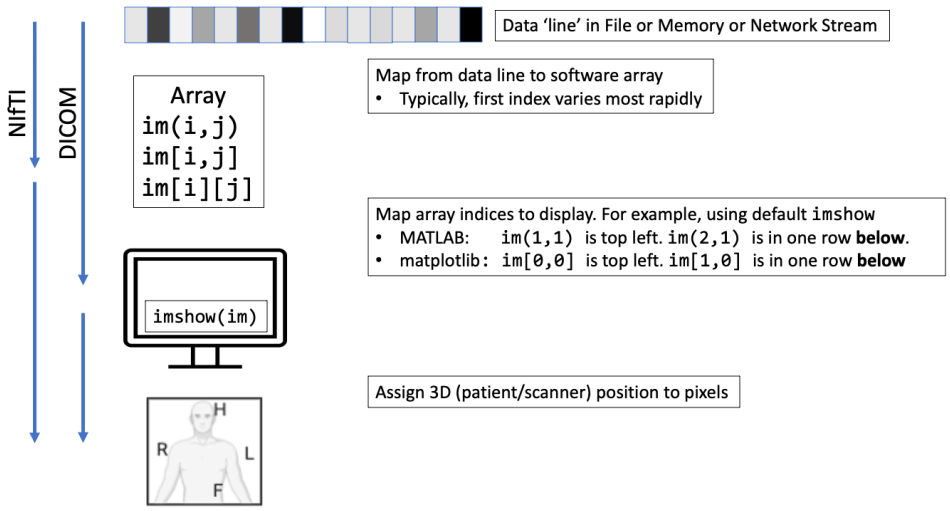

Figure 1: Schematic of the links between acquired data and its 3D location. An array is a mechanism to calculate the position in the 1D line of data corresponding to array indices `i` and `j`. The association of `i` and `j` to image rows and columns is made by the display function - here `imshow`. DICOM specifies that the 1D data line fills the image row by row and contains information on the 3D orientation of the rows and columns. NIfTI specifies that the first array index varies most rapidly when indexing the 1D data line, and a transformation matrix that links array indices to 3D position.

formats.

In the MATLAB default image coordinate system, `imshow(im)` will display an image with the intensity of the top left pixel corresponding to the value in array element `im(1,1)`. The first index corresponds to the row number and this increases as you go down a column of the image. This is similar in Python using 2D `numpy` arrays displayed with `matplotlib.pyplot` using default `extent` and `origin` values except that the array indexing is 0-based. In this case, element `im[0,0]` is at the top left. The method for linking between data and 3D position differs between DICOM and NIfTI.

2 DICOM

The DICOM geometry is depicted in Figure 2. Data in memory should be read in and displayed such that data is placed in the image row by row, starting

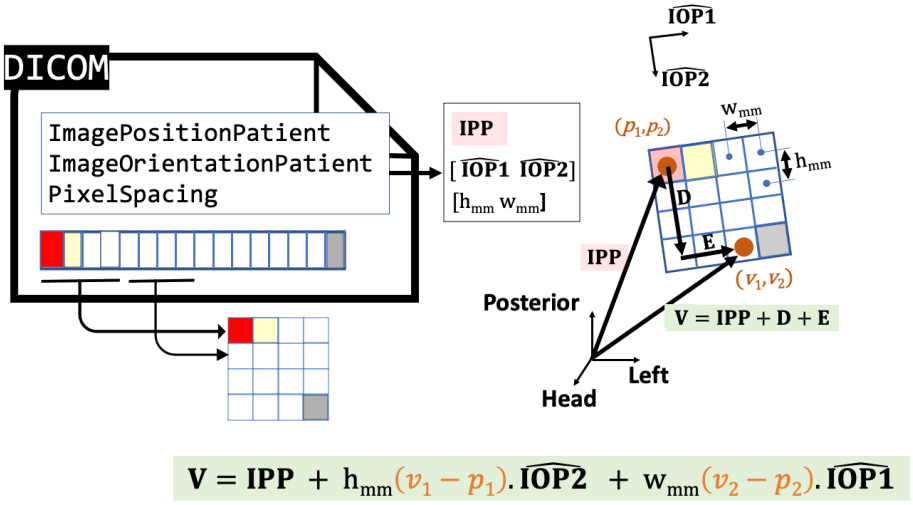

Figure 2: The DICOM scheme. The file contains header parameters and image pixel data. The pixel data is placed row-by-row in the image. The centre of the top left pixel is at **IPP** in patient LPH space and at (p_1, p_2) in image space. A general 2D image point, denoted (v_1, v_2), is at LPH position $\mathbf{V} = \mathbf{IPP} + \mathbf{D} + \mathbf{E}$.

from the top of the image. The `ImagePositionPatient` DICOM entry specifies the 3D position (in LPH+) of the centre of the top left image pixel and `ImageOrientationPatient` specifies the two unit vectors that point along an image row and down a column. Along with `PixelSpacing` (the height and width of the pixels in mm), this is enough information to compute the position in 3D patient-space of any 2D image pixel. For example, the 3D position $\mathbf{V}$ of 2D image pixel at (v_1, v_2) in Figure 2 is just the vector sum

$$\mathbf{V} = \mathbf{IPP} + \mathbf{D} + \mathbf{E} \tag{1}$$

where $\mathbf{D} = \mathrm{h_{mm}}(v_1 - p_1).\widehat{\mathbf{IOP2}}$ and $\mathbf{E} = \mathrm{w_{mm}}(v_2 - p_2).\widehat{\mathbf{IOP1}}$ with terms defined in the Table and the top left image pixel is at image coordinate (p_1, p_2). In MATLAB image coordinates, this is (1,1) and in Python this is (0,0).

Dicom name	Dicom values	Figure representation
`ImagePositionPatient`	$\begin{bmatrix} p_\mathrm{L} & p_\mathrm{P} & p_\mathrm{H} \end{bmatrix}$	**IPP**
`ImageOrientationPatient`	$\begin{bmatrix} a_\mathrm{L} & a_\mathrm{P} & a_\mathrm{H} & b_\mathrm{L} & b_\mathrm{P} & b_\mathrm{H} \end{bmatrix}$	$\begin{bmatrix} \widehat{\mathbf{IOP1}} & \widehat{\mathbf{IOP2}} \end{bmatrix}$
`PixelSpacing`	$\begin{bmatrix} \mathrm{h_{mm}} & \mathrm{w_{mm}} \end{bmatrix}$	

Explicitly writing out the components of these vectors, equation 1 can be

expressed,

$$\begin{bmatrix} v_{\mathrm{L}} \\ v_{\mathrm{P}} \\ v_{\mathrm{H}} \end{bmatrix} = \begin{bmatrix} p_{\mathrm{L}} \\ p_{\mathrm{P}} \\ p_{\mathrm{H}} \end{bmatrix} + \mathrm{h_{mm}}(v_1 - p_1) \begin{bmatrix} b_{\mathrm{L}} \\ b_{\mathrm{P}} \\ b_{\mathrm{H}} \end{bmatrix} + \mathrm{w_{mm}}(v_2 - p_2) \begin{bmatrix} a_{\mathrm{L}} \\ a_{\mathrm{P}} \\ a_{\mathrm{H}} \end{bmatrix}. \tag{2}$$

Using this method of vector addition we can find the 3D patient coordinate $(v_{\mathrm{L}}, v_{\mathrm{P}}, v_{\mathrm{H}})$ of any image point (v_1, v_2). This point can be a pixel centre or a pixel corner (useful for defining patches in 3D plots). The vector addition approach is useful because it avoids the need to explicitly build rotation matrices, deal with quaternions, or consider signs or order of angles. However, it can be convenient to express this transformation using matrices. A more complete expression uses 3D image coordinates but we first use here a 2D image. Equation 3 below uses homogeneous coordinates so that the column vectors on the right-hand side of equation 2 can be combined into one matrix. Note that the first two columns of the matrix in equation 3 are just the second and first vectors of the DICOM `ImageOrientationPatient`. These two vectors describe the orientation of the image plane and thus form the rotation component of the transformation matrix - they are also known as 'direction cosines'. The last column of this matrix is the translational offset `ImagePositionPatient`.

$$\begin{bmatrix} v_{\mathrm{L}} \\ v_{\mathrm{P}} \\ v_{\mathrm{H}} \\ 1 \end{bmatrix} = \begin{bmatrix} b_{\mathrm{L}} & a_{\mathrm{L}} & p_{\mathrm{L}} \\ b_{\mathrm{P}} & a_{\mathrm{P}} & p_{\mathrm{P}} \\ b_{\mathrm{H}} & a_{\mathrm{H}} & p_{\mathrm{H}} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \mathrm{h_{mm}}(v_1 - p_1) \\ \mathrm{w_{mm}}(v_2 - p_2) \\ 1 \end{bmatrix} \tag{3}$$

This can also be written,

$$\begin{bmatrix} v_{\mathrm{L}} \\ v_{\mathrm{P}} \\ v_{\mathrm{H}} \\ 1 \end{bmatrix} = \begin{bmatrix} b_{\mathrm{L}} & a_{\mathrm{L}} & p_{\mathrm{L}} \\ b_{\mathrm{P}} & a_{\mathrm{P}} & p_{\mathrm{P}} \\ b_{\mathrm{H}} & a_{\mathrm{H}} & p_{\mathrm{H}} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \mathrm{h_{mm}} & 0 & 0 \\ 0 & \mathrm{w_{mm}} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -p_1 \\ 0 & 1 & -p_2 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} v_1 \\ v_2 \\ 1 \end{bmatrix} \tag{4}$$

or for later comparison with NIfTI,

$$\begin{bmatrix} v_{\mathrm{L}} \\ v_{\mathrm{P}} \\ v_{\mathrm{H}} \\ 1 \end{bmatrix} = \begin{bmatrix} b_{\mathrm{L}}\mathrm{h_{mm}} & a_{\mathrm{L}}\mathrm{w_{mm}} & c_{\mathrm{L}}\mathrm{s_{mm}} & p_{\mathrm{L}} \\ b_{\mathrm{P}}\mathrm{h_{mm}} & a_{\mathrm{P}}\mathrm{w_{mm}} & c_{\mathrm{P}}\mathrm{s_{mm}} & p_{\mathrm{P}} \\ b_{\mathrm{H}}\mathrm{h_{mm}} & a_{\mathrm{H}}\mathrm{w_{mm}} & c_{\mathrm{H}}\mathrm{s_{mm}} & p_{\mathrm{H}} \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} (v_1 - p_1) \\ (v_2 - p_2) \\ (v_3 - p_3) \\ 1 \end{bmatrix} \tag{5}$$

To handle slices in 3D, the slice normal, or positive through-slice direction, $\hat{\mathbf{C}}$ in LPH is determined from the vector product

$$\hat{\mathbf{C}} = \widehat{\mathbf{IOP1}} \times \widehat{\mathbf{IOP2}} = [\mathbf{a}_{\mathrm{L}}\ \mathbf{a}_{\mathrm{P}}\ \mathbf{a}_{\mathrm{H}}] \times [\mathbf{b}_{\mathrm{L}}\ \mathbf{b}_{\mathrm{P}}\ \mathbf{b}_{\mathrm{H}}]. \tag{6}$$

The distance s_{mm} between slice centres can be found by taking the difference between projections of the `ImagePositionPatient` for adjacent slices onto the slice normal,

$$s_{mm} = |\hat{\mathbf{C}} \bullet \mathbf{IPP_k} - \hat{\mathbf{C}} \bullet \mathbf{IPP_{k+1}}| \tag{7}$$

where $\mathbf{IPP_k}$ and $\mathbf{IPP_{k+1}}$ are the `ImagePositionPatient` values for adjacent slices k and k+1, and $\bullet$ denotes scalar product. Note that the time order in which slices are physically acquired may be different from the positive through-slice direction $\hat{\mathbf{C}}$. The time order can depend on acquisition factors such as bed movements or a need to reduce artifacts from flowing blood or cross-talk. In practice, the only reliable indication of geometrical slice ordering comes from the cross product in Equation 6, and not from DICOM entries such as `SliceNumber`.

In DICOM, there is an `ImagePositionPatient` associated to every 2D slice (or frame) and this can account for non-parallel slices within one file. If when processing regularly spaced volume data, a single `ImagePositionPatient` style vector may be used and then the complete expression for the transformation is given below where the slice pointed to by this single $\mathbf{IPP}$ has index p_3,

$$\begin{bmatrix} v_L \\ v_P \\ v_H \\ 1 \end{bmatrix} = \begin{bmatrix} b_L & a_L & c_L & p_L \\ b_P & a_P & c_P & p_P \\ b_H & a_H & c_H & p_H \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} h_{mm} & 0 & 0 & 0 \\ 0 & w_{mm} & 0 & 0 \\ 0 & 0 & s_{mm} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -p_1 \\ 0 & 1 & 0 & -p_2 \\ 0 & 0 & 1 & -p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 1 \end{bmatrix} \tag{8}$$

Note the order of the columns in the first matrix corresponds to the order of the indices v. For example, increasing the second index v_2 by 1, corresponds to a change in 3D LPH space in the direction of the second column (a_L, a_P, a_H).

3 NIfTI

The NIfTI format is popular in neuroimaging and used by many tools so it is useful to be able to read, write and handle NIfTI format files. The specification of geometrical information in NIfTI has three options [2] but they can each be used to assemble a transformation matrix $\mathbf{M}$. When $\mathbf{M}$ is multiplied by the array indices (i, j, k) it gives the 3D position in a patient RAH+ space (also known as RAS+):

$$\begin{bmatrix} v_R \\ v_A \\ v_H \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} i \\ j \\ k \\ 1 \end{bmatrix}. \tag{9}$$

Note the indices are 0-based, see Figure 3.

Software for reading NIfTI files can often use the NIfTI header data to directly provide the overall transformation matrix $\mathbf{M}$. If the NIfTI `sform` option is

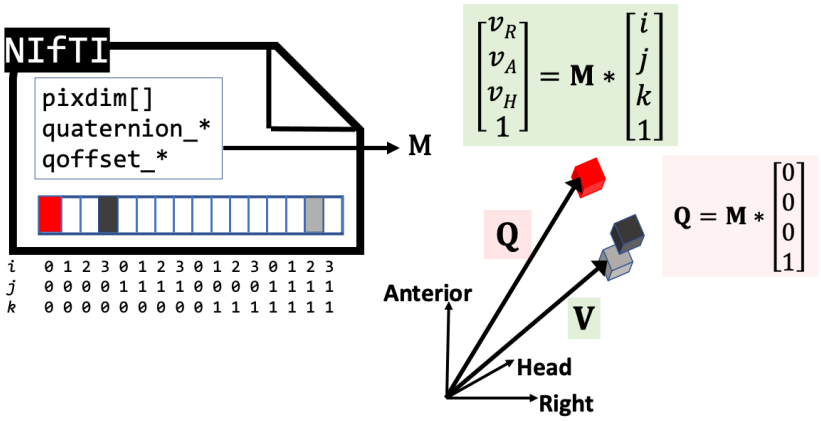

Figure 3: Schematic of the NIfTI format which uses the RAH+ patient system. Data is read into an array such that the first array index i varies most rapidly. The matrix $\mathbf{M}$ is assembled from parameters in the NIfTI header (those used in the `qform` options are shown here). $\mathbf{M}$ can be returned in MATLAB by `niftiinfo` or in the `affine` method of Python's `nibabel`. The 3D location $\mathbf{V}$ of array element (i, j, k) is just the matrix-vector product shown. When the `qform` option is used, the voxel with index $(0, 0, 0)$ is at position $\mathbf{Q}$ whose elements are just the `qoffset_x/y/z` values in the header.

used, $\mathbf{M}$ need not be affine. If the `qform` option is used, $\mathbf{M}$ is a composition of a rotation component determined from a quaternion, an offset (similar to `ImagePositionPatient` in DICOM) and a scaling for the pixel size:

$$\mathbf{M} = \begin{bmatrix} R11 & R12 & R13 & \texttt{qoffset\_x} \\ R21 & R22 & R23 & \texttt{qoffset\_y} \\ R31 & R32 & R33 & \texttt{qoffset\_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \texttt{pixdim[1]} & 0 & 0 & 0 \\ 0 & \texttt{pixdim[2]} & 0 & 0 \\ 0 & 0 & \texttt{qfac*pixdim[3]} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

where $R11$ etc are computed from the quaternion in the header, `qfac` = `pixdim[0]` and the voxel sizes are in the next three elements of `pixdim`.

$$\mathbf{M} = \begin{bmatrix} R11 * \texttt{pixdim[1]} & R12 * \texttt{pixdim[2]} & R13 * \texttt{qfac*pixdim[3]} & \texttt{qoffset\_x} \\ R21 * \texttt{pixdim[1]} & R22 * \texttt{pixdim[2]} & R23 * \texttt{qfac*pixdim[3]} & \texttt{qoffset\_y} \\ R31 * \texttt{pixdim[1]} & R32 * \texttt{pixdim[2]} & R33 * \texttt{qfac*pixdim[3]} & \texttt{qoffset\_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

Taking into account that NIfTI uses RAH+, whereas DICOM uses LPH+, and comparing with the matrix in equation 5, the elements of $\mathbf{M}$ correspond to DICOM terms as follows:

$$\begin{bmatrix} -b_{\mathrm{L}}\mathrm{h}_{\mathrm{mm}} & -a_{\mathrm{L}}\mathrm{w}_{\mathrm{mm}} & -c_{\mathrm{L}}\mathrm{s}_{\mathrm{mm}} & -p_{\mathrm{L}} \\ -b_{\mathrm{P}}\mathrm{h}_{\mathrm{mm}} & -a_{\mathrm{P}}\mathrm{w}_{\mathrm{mm}} & -c_{\mathrm{P}}\mathrm{s}_{\mathrm{mm}} & -p_{\mathrm{P}} \\ b_{\mathrm{H}}\mathrm{h}_{\mathrm{mm}} & a_{\mathrm{H}}\mathrm{w}_{\mathrm{mm}} & c_{\mathrm{H}}\mathrm{s}_{\mathrm{mm}} & p_{\mathrm{H}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4 MATLAB

In the MATLAB default image coordinate system, array `im` is indexed as `im(r,c)` where `r` denotes row number and `c` column number. Row numbers increase going *down* the image. The centre of the top left pixel corresponds to the pixel indexed as `im(1,1)` and is assigned the coordinate (1,1). The distance between neighbouring pixels is 1. Pixel centres coincide with integer coordinates. The edge of the top left corner is at (0.5, 0.5). For 3D arrays in MATLAB, the third dimension is interpreted as the slice and (v_1, v_2, v_3) corresponds to (row, column, slice). The slice pointed to by a single `ImagePositionPatient` or `qoffset` can be given a coordinate value of 1 so that $(p_1, p_2, p_3) = (1, 1, 1)$.

MATLAB reads data from a file filling the first index first ('Fortran order'). This corresponds to the NIfTI standard. When displayed, this first index corresponds to row number in MATLAB, meaning the image is filled column by

column. However DICOM requires the image to be filled row by row so internally `dicomread` in MATLAB is expected to perform a transpose upon reading.

5 PYTHON

Using `imshow` from `matplotlib.pyplot` for display, with default `extent` and `origin` values, then for a 2D array the display is the same as MATLAB except that indices start at 0 and not 1, i.e. (row, column) order with unit spacing. NIfTI specifies that the mapping from array index to the 1D data line should use Fortran order where the first index varies most rapidly. This is the opposite of the default behaviour for `numpy` arrays which by default use C ordering. The `nibabel` package can read NIfTIs into 3D `numpy` arrays with Fortran ordering. The ordering can be determined from the `flags` method.

6 Comments

DICOM and NIfTI displays of the same data may present visually with different on-screen orientations, however, these are consistent if the associated geometrical information is correct. This can be confirmed by labelling the image edges with the patient directions.

The DICOM way of describing geometry (with a point, two directions and a scale), is relatively easy to update if the image is manipulated, for example, after cropping, padding, rotation or flip for display purposes. The display convention in radiology is shown in Figure 4. Note that in some neurology applications, a different convention is used with the viewing position being from behind the patient.

Flipping or permuting of data arrays is usually only necessary in limited situations such as reading and writing to files or as a final step to change appearance to correspond to radiological conventions. A programmer should think twice before flipping or permuting and be able to justify the reason.

It can help the programmer keep the geometry information consistent by considering the more general interpretation of the transformation matrix. In a transformation matrix, the rows correspond to the output space i.e. LPH if DICOM or RAH if NIfTI, and the column order corresponds to order of the image array indices. For an image coordinate (v_1, v_2, v_3) and LPH patient-space,

the matrix transformation is,

$$\begin{bmatrix} v_{\mathrm{L}} \\ v_{\mathrm{P}} \\ v_{\mathrm{H}} \\ 1 \end{bmatrix} = \begin{bmatrix} d1_{\mathrm{L}} & d2_{\mathrm{L}} & d3_{\mathrm{L}} & p_{\mathrm{L}} \\ d1_{\mathrm{P}} & d2_{\mathrm{P}} & d3_{\mathrm{P}} & p_{\mathrm{P}} \\ d1_{\mathrm{H}} & d2_{\mathrm{H}} & d3_{\mathrm{H}} & p_{\mathrm{H}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathrm{s1}_{\mathrm{mm}} & 0 & 0 & 0 \\ 0 & \mathrm{s2}_{\mathrm{mm}} & 0 & 0 \\ 0 & 0 & \mathrm{s3}_{\mathrm{mm}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} v_1 - p_1 \\ v_2 - p_2 \\ v_3 - p_3 \\ 1 \end{bmatrix}$$

where (p_1, p_2, p_3) is the image coordinate of the voxel at the patient location $(p_{\mathrm{L}}, p_{\mathrm{P}}, p_{\mathrm{H}})$. If the value of the index v_1 in the first image array dimension is increased by 1, that corresponds to a move of $\mathrm{s1}_{\mathrm{mm}}$ in the direction of unit vector $(d1_{\mathrm{L}}, d1_{\mathrm{P}}, d1_{\mathrm{H}})$ (the first column). Similarly for the other image array dimensions - the order of the matrix columns should match the array indices.

A common cause of programming confusion is to associate physical directions to array dimensions. A useful thought process is to first think of arrays as just something to convert indices representing multiple dimensions to a position in a 1D line of data. Then try to just deal with indices as first dimension, second dimension etc and not 'x,y,z'. Finally at the point of display, the programmer needs to be aware of the mapping from index to image row or column.

The DICOM `ImagePositionPatient` used here, and in modern systems, points to the *centre* of the top left pixel. Some legacy systems may assume it points to the corner edge.

In DICOM, the images, or frames, are 2D with no strong concept of through-slice direction or slice ordering. This allows for acquiring slices that are not parallel (for example following the curvature of the spine) or in any temporal order. A separate `ImagePositionPatient` is provided for every 2D image frame. This is true even in Enhanced DICOM where all slices of one acquisition are stored in a single file, within this file there will be an `ImagePositionPatient` for every slice. By contrast in NIfTI, there is just one `qoffset` vector for the volume. If using 3D image coordinates after reading from DICOM, the vector $(p_{\mathrm{L}}, p_{\mathrm{P}}, p_{\mathrm{H}})$, should correspond to the same slice as image coordinate p_3. In essence, the `ImagePositionPatient` for one of the slices is chosen as the reference point.

If a user wants to assemble frames into a 3D array that is right-handed, the user needs to compute the positive through-slice direction $\hat{\mathbf{C}}$ and sort frames using the projection $\hat{\mathbf{C}} \bullet \mathbf{IPP}_{\mathrm{frame}}$. The user should not rely on the DICOM `SliceNumber` to infer geometric position. Acquired slices may have gaps between, or overlap, so the DICOM `SliceThickness` should not be used as the distance between successive slice centres. Instead, calculate s_{mm} as in equation 7.

The location of the DICOM origin is not specified, though DICOM images with the same `FrameOfReferenceUID` all use the same origin and can be viewed and manipulated consistently. In MRI, a patient is 'landmarked' and the bed

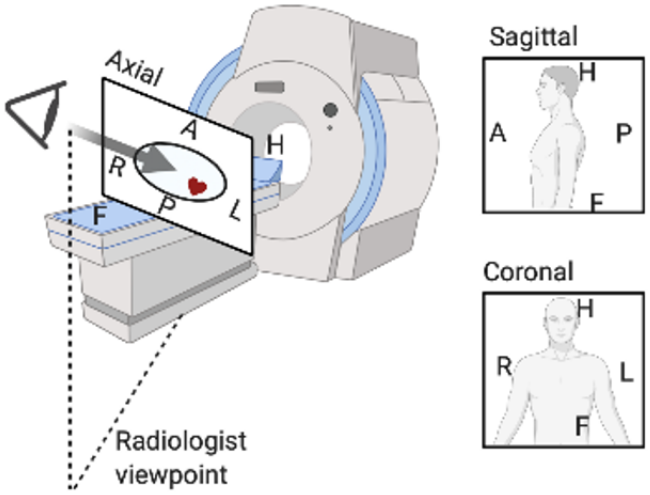

Figure 4: The radiological convention for image display is similar to viewing a patient as shown (assuming here the patient is on their back and head-first into the scanner). Figure created using BioRendr.com

moved so that the landmarked point initially goes to the magnet isocentre. During a session, the bed may move to optimise image quality and this will be accounted for by the scanner - the DICOM origin will remain fixed relative to the patient but the origin is not necessarily at the scanner isocentre. The `FrameOfReferenceUID` is a unique number for the duration the patient is in the scanner, it changes for each subject and if the patient is repositioned (because the geometrical link between scans is lost).

An MR scanner can only output correct LPH+ patient coordinates if it has been told the correct patient orientation in the scanner e.g. head or feet first, prone or supine. This relies on the operator.

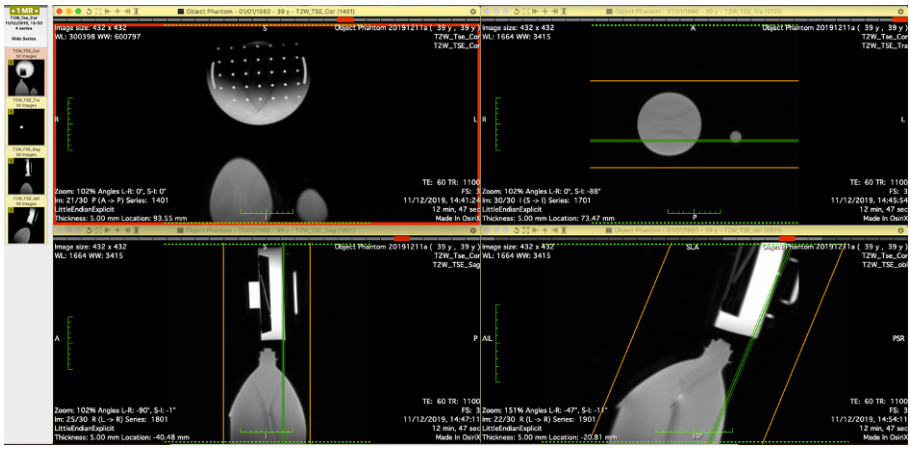

Figure 5: Screenshot from Horos of the four acquisitions. Arrangement was a structured phantom flat on the bed as a 'head', a large bottle as a right 'leg' and a small bottle as a left 'leg' for a head-first, supine patient. There are three scans in almost orthogonal planes, and one oblique scan. Note imperfect warp correction in the oblique scan. The green lines show the slice position for the top left pane (coronal). Transverse (axial) scan is top right and sagittal bottom left.

7 Supporting Code and Data

Supporting code and data is provided [5] from four volume acquisitions at different orientations of a static arrangement of phantoms as shown in Figure 5. The scans were acquired on a Philips Ingenia MR scanner (software version 5.4.1) and saved as DICOM, NIfTI and NIfTI (FSL). MATLAB `geomdemo.m` and Python `geomdemo.py` are similar and plot the slice projections $\hat{\mathbf{C}} \bullet \mathbf{IPP}$ for each frame (see Figure 6), 3D plots of representative intersecting slices with their coordinates determined using two methods - by vector addition and by constructing transformation matrices (Figure 7), and, single slices from each acquisition. MATLAB script `ndcomp.m` compares the transformation matrices from the DICOM and NIfTI files and uses `ori_info.m` to label images with the patient direction - see Figure 8. The code is written for explanation and not speed or memory use. For further software and associated descriptions, see [3] and the Wiki page for `dcm2niix` [4].

References

[1] The DICOM Specification `https://www.dicomstandard.org`

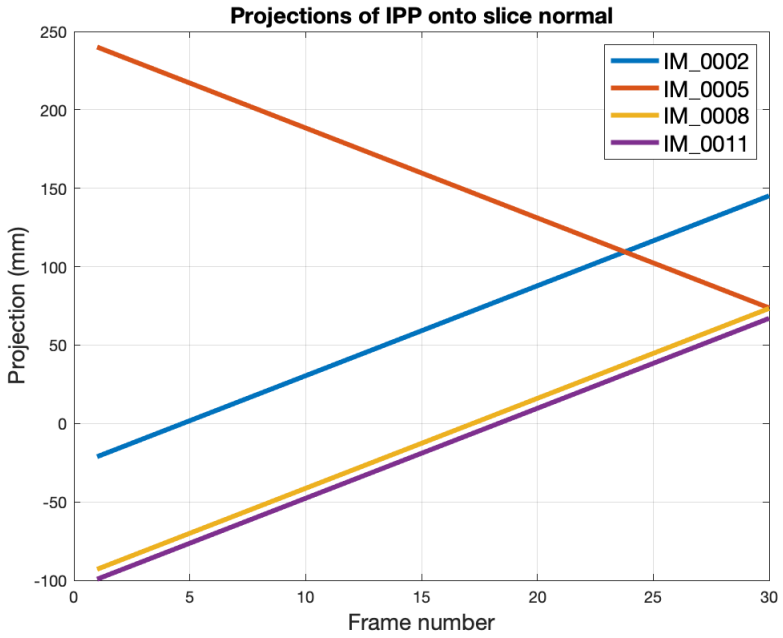

Figure 6: Output from supplied code. Projections of `ImagePositionPatient`s onto the slice normals. Note that for the transverse scan (IM_0005), the slice order is not along the positive slice normal direction, suggesting the data should be re-sorted if it is to be treated as a volume. Code comments in the supplied MATLAB file provide more details.

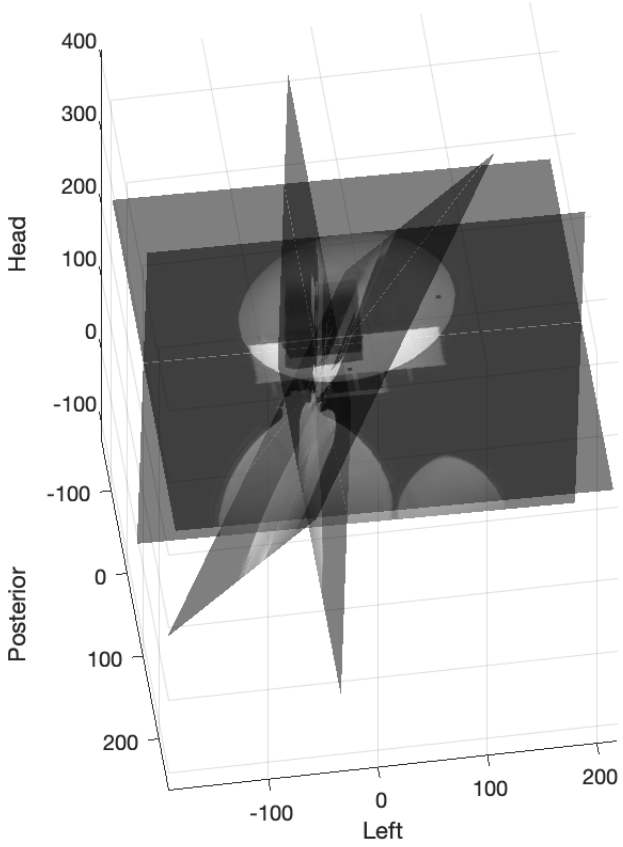

Figure 7: Output from supplied code. 3D plot showing output from supplied code for selected slices in LPH space. Using either the vector addition or transformation matrix methods give the same figures. Spatial alignment can be checked visually by rotating the figure on screen and checking the positions of the bottle edges coincide in different slices.

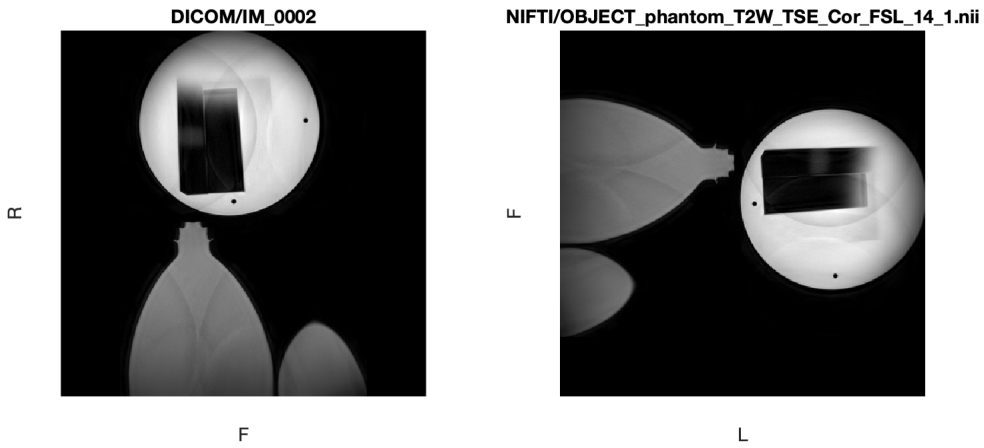

Figure 8: Output from `ndcomp.m` showing DICOM and NIfTI images from the same acquisition. Labelling the image edges shows that despite the different appearances the geometrical information is consistent (large bottle is to right, structured phantom towards head).

[2] The NIfTI specification `https://nifti.nimh.nih.gov/pub/dist/src/niftilib/nifti1.h`

[3] Li et al *The first step for neuroimaging data analysis: DICOM to NIfTI conversion*. J Neurosci Methods. (2016) 1;264:47-56. doi: 10.1016/j.jneumeth.2016.03.001.

[4] The `dcm2niix` Wiki: `https://www.nitrc.org/plugins/mwiki/index.php/dcm2nii:MainPage`

[5] Zenodo location of code and data: `https://doi.org/10.5281/zenodo.6467772`