HOWTO: `vox2ras` calculation from `meas.asc`

Rudolph Pienaar

9th June 2004

Abstract

The `vox2ras` matrix defines the rotational and translational mapping between logical and real-world volume image coordinates. It is an important parameter in the visualization of reconstructed scanner data. In the case of on-line image reconstruction, it is usually determined from DICOM headers. For off-line image reconstruction, however, only the raw k-space scanner data and a brief header file are available. The header file, `meas.asc`, does not by default contain enough information to uniquely determine the `vox2ras` matrix.

This technical report presents two methods by which to retrieve the `vox2ras` data using the `meas.asc` file. The first method relies on embedding the rotational component directly in the `meas.asc` file during a scan. The second method attempts to solve for the `vox2ras` rotational component using only a single direction cosine vector and some experimental data. The translational component of the `vox2ras` matrix is calculated in both cases once the rotational data has been determined. The overall accuracy of the rotational and translational calculations are discussed in the results section, which shows that an accuracy to about four decimal places can be achieved with these methods.

1 Introduction

In-house visualization of reconstructed scanner volumes using tools like `tkmedit` [2] rely on a `vox2ras` matrix that specifies the rotational and translational mapping between logical scanner and real-world volume coordinates. Typically, k-space scanner data is reconstructed on-line, i.e. during the scanning session, and resultant image volumes are saved in the scanner database in DICOM [1] format. Various tools have been developed that analyze these DICOM reconstructed volumes and determine an appropriate `vox2ras` matrix.

Sometimes it is more advantageous to defer image volume reconstruction off-line, in which case only scanner raw k-space data is collected and no visualization is performed at the scanner console. Such a scenario often arises when on-line reconstruction of an image volume at the scanner console might take a prohibitively long time, particularly in multi-echo, multi-channel, and/or high resolution scans. For such off-line collected data, corresponding DICOMs (and hence `vox2ras` data) might not exist, and the only data present might be the actual raw k-space data itself (saved in Siemens scanners in a file called `meas.out`) and a small descriptive header file (in the Siemens case saved in a file called `meas.asc`).

This technical report describes some techniques for calculating a `vox2ras` matrix for these off-line data volumes using only `meas.asc`. By implicit assumption, the scope of this report is limited to Siemens scanners and their associated raw data files. Two methods of determining the rotational components of the `vox2ras` matrix will be discussed: (1) a direct parsing of the `meas.asc` file for sequences that have been appended to embed a scanner-internal rotation matrix directly in `meas.asc`; and (2) a method to calculate the rotational components for cases that do not have this embedded scanner-internal rotational matrix appended to `meas.asc`. The first method depends on non-standard changes to the Siemens scan techniques, and is only available to sites that have an IDEA development site license. Although more accurate, as a technique it suffers from being non-portable. The second method uses only the standard Siemens `meas.asc` header file, and while slightly less accurate, is portable across any Siemens scanner. Once the rotational components of the `vox2ras` matrix have been determined, we can calculate the translational component that defines the center of k-space in the reconstructed image volume.

After discussing the methods, some brief results will be presented that compare the accuracy of each technique against some reference data. This reference data is collected from an analysis of ten separate scan sessions for which embedded scanner-rotation matrix data *and* DICOM derived `vox2ras` information is available.

2 Methods

2.1 The `vox2ras` matrix

A `vox2ras` matrix has the following structure:

$$\mathbf{V} = \begin{bmatrix} x_r & y_r & z_r & c_r \\ x_a & y_a & z_a & c_a \\ x_s & y_s & z_s & c_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

or, alternatively

$$\mathbf{V} = \begin{bmatrix} [\mathbf{x}] & [\mathbf{y}] & [\mathbf{z}] & [\mathbf{c}] \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, and $\mathbf{c}$ are the constituent 3x1 vectors. Direction cosines defining the orientation of the image volume in real space are encoded in the $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ vectors. These vectors describe the mapping from a logical coordinate system to a physical RAS (Right-Anterior-Superior) sense. The $\mathbf{x}$ vector describes how the "`PhaseEncode`" logical dimension is mapped to RAS coordinates; the $\mathbf{y}$ vector describes how the "`ReadOut`" logical dimension is mapped; and the $\mathbf{z}$ vector describes how the "`SliceSelect`" logical dimension is mapped.

The RAS (Right-Anterior-Superior) sense is patient-relative, and specifies the following orthogonal base axes originating in the center of k-space, i.e. the center of the scanned volume: (1) from the origin to the patient's right; (2) from the origin to the patient's front; and (3) from the origin towards the top of the patient's head. These directions are normal to the following planes respectively: (1) sagittal; (2) coronal; (3) transverse.

Additionally, the $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ vectors are orthogonal (but not orthonormal) and are scaled by the real-space voxel dimension size $\mathbf{d} = \begin{bmatrix} d_x & d_y & d_z \end{bmatrix}$ such that

$$\begin{bmatrix} x_r & y_r & z_r \\ x_a & y_a & z_a \\ x_s & y_s & z_s \end{bmatrix} = \begin{bmatrix} \hat{x}_r & \hat{y}_r & \hat{z}_r \\ \hat{x}_a & \hat{y}_a & \hat{z}_a \\ \hat{x}_s & \hat{y}_s & \hat{z}_s \end{bmatrix} \begin{bmatrix} d_x & 0 & 0 \\ 0 & d_y & 0 \\ 0 & 0 & d_z \end{bmatrix} \tag{3}$$

where the $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$ vectors on the right hand side are normalized.

2.2 Scanner coordinates senses and internal rotation matrix

Siemens scanners operate in a right-handed orthogonal LPS (Left-Posterior-Superior) sense [7]. An LPS sense relates to an RAS sense by inverting the directions of the first two dimensions

$$\text{LPS}=(\text{-R})(\text{-A})\text{S} \tag{4}$$

This inverted direction of the sagittal and coronal planes implies that volumes reconstructed in an LPS sense need to invert the save directions of their sagittal and coronal dimensions to be correctly viewed in RAS sense tools. Although beyond the scope of this report, this LPS to RAS relation has direct implications for any off-line image reconstruction software that wishes to save image volumes in an RAS sense (the in-house developed MGH format assumes that images are encoded in an RAS sense).

The logical scanner coordinate system is fixed by the magnetic gradient directions $\mathbf{G}$ that define the "`PhaseEncode`", "`ReadOut`", and "`SliceSelect`" directions. These are related according to [7]

$$G_{\text{PhaseEncode}} \times G_{\text{ReadOut}} = G_{\text{SliceSelect}} \tag{5}$$

The fixed xyz-coordinate system of the scanner hardware is defined by standing in front on the scanner and peering straight ahead into the scanner bore. The x-axis points from left to right, the y-axis points down

Algorithm 1 ICE programming snippets to capture the Siemens internal $\mathbf{R}$ rotation matrix.

```
bool IceProgramADCRotSave::prepare () {
    // Open meas.asc and meas.out for access
}
bool IceProgramADCRotSave::online (MdhProxy &aMdh, IceAsFifo &fifoAs) {
    // Rotation matrix data
    sSliceData s_SD; double adRM[3][3];
    s_SD = aMdhCopy.getSliceData(); aMdhCopy.getRotMatrix(adRotMatrix, s_SD);
    for(int i=0; i<3; i++)
        fprintf(meas.asc,
                "### adRM[i][0] = %f adRM[i][1] = %f adRM[i][2] = %f\n",
                adRM[i][0], adRM[i][1], adRM[i][2]);
}
```

Algorithm 2 Siemens rotation matrix written to a non-standard `meas.asc`.

```
### Additional derived data
###
### Bandwidth_per_pixel_for_ADC[0] = 651.042
### adRM[0][0] = 0.0367939 adRM[0][1] = 0.9924 adRM[0][2] = 0.117422
### adRM[1][0] = -0.0270481 adRM[1][1] = -0.11647 adRM[1][2] = 0.992826
### adRM[2][0] = 0.998957 adRM[2][1] = -0.039706 adRM[2][2] = 0.0225572
```

to up, and the z-axis points from the back of the bore straight over the patient table towards your viewpoint. If one visualizes a patient lying head first, and on their back on the table, we can see that the from the patient's perspective a point in the scanner xyz space relates in the following manner to the patient's RAS sense

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} R \\ A \\ S \end{bmatrix} \tag{6}$$

and this xyz coordinate system is thus orientated normal to the patient's sagittal, coronal, and and transverse planes respectively.

2.3 Direct parsing of rotational component

The scanner keeps track of a mapping between its logical coordinate system and its fixed physical system in an interal rotation matrix structure, $\mathbf{R}$ such that

$$\begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix} = \mathbf{R} \begin{bmatrix} G_{\text{PhaseEncode}} \\ G_{\text{ReadOut}} \\ G_{\text{SliceSelect}} \end{bmatrix} \tag{7}$$

where $\mathbf{R}$ is defined in an LPS sense. This $\mathbf{R}$ matrix can quite easily be manipulated into a `vox2ras` matrix; indeed the challenge as such becomes a case of capturing the $\mathbf{R}$ matrix during a scan and recording it in a manner that is accessible to an off-line reconstruction program.

A workable solution is to adapt the sequence C++ generating code. Within the Siemens ICE [9] development environment a candidate ICE program associated with a given sequence can have its `::online(...)` method adapted so as to record the $\mathbf{R}$ matrix[1]. This matrix is recorded at the end of a `meas.asc` file and marked with comment characters to protect it from hidden side effects that might arise. The core concepts of capturing the $\mathbf{R}$ matrix are presented in Algorithm 1, and an example of an $\mathbf{R}$ matrix appended to a `meas.asc` file is given in Algorithm 2. This matrix can then be parsed from the `meas.asc` in a straightforward manner by reconstruction software.

[1]ICE programming concepts are beyond the scope of this report.

Algorithm 3 An example of standard sagittal direction cosine fields in `meas.asc`.

```
...
sSliceArray.asSlice[0].sNormal.dSag = 0.998068
sSliceArray.asSlice[0].sNormal.dCor = -0.023969
sSliceArray.asSlice[0].sNormal.dTra = 0.057326
...
```

An additional comment needs to be made. Once the code was developed and deployed it was noticed that the Siemens matrix is row dominant (evident in Equation 7), while the `vox2ras` matrix is column dominant (Equation 1). This slight oversight is easily corrected by simply transposing the **R** matrix. Given this information, we can specify the 3x3 rotational component of the `vox2ras` matrix in terms of **R** as

$$\begin{aligned}
\mathbf{V}_{\text{rot}} &= \mathbf{X}_1\mathbf{R}^{\mathbf{T}}\mathbf{X}_2\mathbf{D} \\
\mathbf{X}_1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \\
\mathbf{X}_2 &= \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \\
\mathbf{D} &= \begin{bmatrix} d_x & 0 & 0 \\ 0 & d_y & 0 \\ 0 & 0 & d_z \end{bmatrix}
\end{aligned} \tag{8}$$

where $\mathbf{X}_1$ and $\mathbf{X}_2$ implement an LPS to RAS conversion (with the added stipulation that the sagittal plane is orientated such that the "nose" points to the "right" of the view window). The voxel dimension matrix, **D**, contains the size of each voxel as described in Section 2.1.

2.4 Indirect calculation of rotational component

The direct parsing of the scanner-based rotation matrix is the most efficient mechanism by which to determine the rotational component of the `vox2ras` matrix. This scanner matrix, however, is a non-standard addition to the `meas.asc` file and thus may not be available for all scans or under all situations. In order to address this shortcoming, we will now discuss a mechanism for calculating the rotational component of the `vox2ras` matrix using only the standard `meas.asc` file.

By default, the `meas.asc` captures a vector orthonormal to the main slab orientation, i.e the direction cosine for the `SliceSelect` plane. This vector is defined in a (sagittal, coronal, transverse) coordinate system – see Algorithm 3 for an example. Referring to Equation 2, this vector is simply the normalized **z** vector of the `vox2ras` matrix. Siemens defines the two additional orthogonal vectors (the two remaining direction cosines, i.e. the `PhaseEncode` direction and the `ReadOut` direction) by rotating the SliceSelect vector by 90 degrees and then calculating the PhaseEncode reference vector as shown in Algorithm 4.

From Algorithm 4, we see that the `PhaseEncode` vector is defined for each main orientation of the `SliceSelect`. In the case when the `SliceSelect` is primarily transverse, the reference `PhaseEncode` vector is

$$\mathbf{x}_{\text{reference}} = \begin{bmatrix} 0 \\ z_s\sqrt{z_a^2 + z_s^2}^{\,-\frac{1}{2}} \\ -z_r\sqrt{z_a^2 + z_s^2}^{\,-\frac{1}{2}} \end{bmatrix} \tag{9}$$

while if the `SliceSelect` is primarily coronal, the reference `PhaseEncode` vector is

$$\mathbf{x}_{\text{reference}} = \begin{bmatrix} z_a\sqrt{z_r^2 + z_a^2}^{\,-\frac{1}{2}} \\ -z_r\sqrt{z_r^2 + z_a^2}^{\,-\frac{1}{2}} \\ 0 \end{bmatrix} \tag{10}$$

Algorithm 4 Defining the `PhaseEncode` reference vector from the `SliceSelect` vector. `Vc_P` denotes the `PhaseEncode` reference; `Vc_Cn` denotes the `SliceSelect` (in [5] derived from [8]).

```
%% phase reference vector
Vc_P = zeros(3, 1);
switch ch_orientation
    case 't'
        Vc_P(1) =  0;
        Vc_P(2) =  Vc_Cn(3)*sqrt(1/(Vc_Cn(2)*Vc_Cn(2)+Vc_Cn(3)*Vc_Cn(3)));
        Vc_P(3) = -Vc_Cn(2)*sqrt(1/(Vc_Cn(2)*Vc_Cn(2)+Vc_Cn(3)*Vc_Cn(3)));
    case 'c'
        Vc_P(1) =  Vc_Cn(2)*sqrt(1/(Vc_Cn(1)*Vc_Cn(1)+Vc_Cn(2)*Vc_Cn(2)));
        Vc_P(2) = -Vc_Cn(1)*sqrt(1/(Vc_Cn(1)*Vc_Cn(1)+Vc_Cn(2)*Vc_Cn(2)));
        Vc_P(3) =  0;
    case 's'
        Vc_P(1) = -Vc_Cn(2)*sqrt(1/(Vc_Cn(1)*Vc_Cn(1)+Vc_Cn(2)*Vc_Cn(2)));
        Vc_P(2) =  Vc_Cn(1)*sqrt(1/(Vc_Cn(1)*Vc_Cn(1)+Vc_Cn(2)*Vc_Cn(2)));
        Vc_P(3) =  0;
    otherwise
        fprintf(1, 'Unknown orientation parameter passed.'  );
    return;
end
```

and for the case when the `SliceSelect` is primarily sagittal, the reference `PhaseEncode` vector is

$$\mathbf{x}_{\text{reference}} = \begin{bmatrix} -z_a \sqrt{z_r^2 + z_a^2}^{-\frac{1}{2}} \\ z_r \sqrt{z_r^2 + z_a^2}^{-\frac{1}{2}} \\ 0 \end{bmatrix} \tag{11}$$

Having thus defined the `PhaseEncode` reference $\mathbf{x}_{\text{reference}}$ vector, we calculate the `ReadOut` reference vector $\mathbf{y}_{\text{reference}}$ as the cross product between the `SliceSelect` and the `PhaseEncode`,

$$\mathbf{y}_{\text{reference}} = \mathbf{z}_{\text{meas.asc}} \times \mathbf{x}_{\text{reference}} \tag{12}$$

We have now defined the direction cosines in a reference frame. To arrive at the rotational `vox2ras` matrix, we simply rotate the $\mathbf{x}_{\text{referece}}\mathbf{y}_{\text{reference}}$ plane about $\mathbf{z}_{\text{meas.asc}}$ by the radian amount defined in the `sSliceArray.asSlice[0].dInPlaneRot` (or θ_{rot}) field of `meas.asc`. Combining all of this together, we have

$$\begin{aligned}
\mathbf{V}_{\text{rot}} &= \begin{bmatrix} [\mathbf{x}_{\text{reference}}] & [\mathbf{y}_{\text{reference}}] & [\mathbf{z}_{\text{meas.asc}}] \end{bmatrix} \begin{bmatrix} \cos\theta_{\text{rot}} & \sin\theta_{\text{rot}} & 0 \\ -\sin\theta_{\text{rot}} & \cos\theta_{\text{rot}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{XD} \\
\mathbf{X} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
\mathbf{D} &= \begin{bmatrix} d_x & 0 & 0 \\ 0 & d_y & 0 \\ 0 & 0 & d_z \end{bmatrix}
\end{aligned} \tag{13}$$

with **X** and **D** as described in Section 2.3. Note that an implementation of calculating the rotational component of the `vox2ras` matrix is prototyped in [5].

Algorithm 5 An example of slice position $\mathbf{c}_s = \begin{bmatrix} c_{sr} & c_{sa} & c_{ss} \end{bmatrix}^T$ in real xyz space, defined in `meas.asc`.

```
...
sSliceArray.asSlice[0].sPosition.dSag = 2.419566
sSliceArray.asSlice[0].sPosition.dCor = -22.07259
sSliceArray.asSlice[0].sPosition.dTra = 4.0306
...
```

2.5 Center of k-space calculation

The previous two Sections concerned themselves with determining the rotational component of the `vox2ras` matrix, i.e. the constituent **x**, **y**, and **z** vectors. The remaining **c** vector, i.e. the offset to the center of k-space still remains. As with the indirect calculation of the rotation matrix, we start by parsing the `meas.asc` file – this time for the values that specify the position of the first slice in real scanner xyz space (see Algorithm 5).

Our intuition would suggest that given the slice position $\mathbf{c}_s$, we can find the center of k-space by traveling along the negative `SliceSelect` vector for a distance equal to one-half of the image volume. However, after studying several sequences for which raw data had been acquired, and for which `vox2ras` matrices were available, the following method of finding the k-space center was found:

First, calculate two end point vectors, $\mathbf{n} = \begin{pmatrix} n_r & n_a & n_s \end{pmatrix}$ and $\mathbf{p} = \begin{pmatrix} p_r & p_a & p_s \end{pmatrix}$ such that

$$\begin{aligned} \mathbf{n} &= -\mathbf{c}_s - \frac{1}{2}\Delta_{\mathrm{PE}}\mathbf{x} - \frac{1}{2}\Delta_{\mathrm{RO}}\mathbf{y} - \frac{1}{2}\Delta_{\mathrm{SS}}\mathbf{z} \\ \mathbf{p} &= +\mathbf{c}_s - \frac{1}{2}\Delta_{\mathrm{PE}}\mathbf{x} - \frac{1}{2}\Delta_{\mathrm{RO}}\mathbf{y} - \frac{1}{2}\Delta_{\mathrm{SS}}\mathbf{z} \end{aligned} \tag{14}$$

where $\mathbf{\Delta} = \begin{bmatrix} \Delta_{\mathrm{PE}} & \Delta_{\mathrm{RO}} & \Delta_{\mathrm{SS}} \end{bmatrix}$ are the logical lengths of the `PhaseEncode`, `ReadOut`, and `SliceSelect` dimensions. Note the inverted signs of the first two positional components. Given these two vectors, the final k-space center is

$$\mathbf{c} = \begin{bmatrix} n_r + d \\ n_a \\ p_s \end{bmatrix} \tag{15}$$

Note the value d that is added to the c_r component. It would appear that there is a consistent error of about one half pixel in the `PhaseEncode` k-space center in all the scans that were analyzed. For the standard GLEEK sequences at the NMR center of $\Delta = \begin{bmatrix} 256 & 256 & 128 \end{bmatrix}$and voxel dimensions of $\begin{bmatrix} 1 & 1 & 1.33 \end{bmatrix}$mm, $d = -0.67$.

Note that an implementation of this method is prototyped in [4].

3 Results

The results section is concerned mainly with the accuracy of the methods described in this paper. To this end, we will compare the `vox2ras` matrices derived from the methods of Sections 2.3, 2.4, and 2.5. We will assume that we can compare our results against a corresponding `vox2ras` matrix derived from DICOM headers. We will also assume that this DICOM-derived `vox2ras` is the “correct” matrix.

To this end, we will examine the rotational submatrix and center of k-space vector separately for both the direct parsing and the indirect calculation approaches. In the case of the rotational component, we will define a comparison matrix as

$$\mathbf{V}_{\mathrm{rotComparison}} = \mathbf{V}_{\mathrm{rotDICOM}}\mathbf{V}_{\mathrm{rot}}^{-1} \tag{16}$$

where $\mathbf{V}_{\mathrm{rot}}$ is determined first by direct parsing, and then by indirect calculation. A root-mean-square of the comparison matrix is then evaluated

Table 1: rms values for each of the methods used in this paper. The first column denotes the `inPlaneRot` for a given sample case. For columns two through five, an exact value of 1 would indicate complete correlation between the DICOM reference vox2ras and the method used for that column. The last row in *italic* face denotes the mean value of each column. DP = Direct Parsing; IC = Indirect Calculation

inPlaneRot (rad)	$\mathrm{rms}_{\mathrm{rot}}$ DICOM/DP	$\mathrm{rms}_{\mathrm{rot}}$ DICOM/IC	rms c DICOM/DP	rms c DICOM/IC
3.4560473e-03	1.0000443e+00	1.0000443e+00	1.0000896e+00	1.0000894e+00
-2.1622696e-01	1.0000434e+00	1.0000434e+00	1.0003922e+00	1.0003921e+00
-4.8874619e-02	1.0000302e+00	1.0000302e+00	1.0000161e+00	1.0000160e+00
-2.3524211e-02	1.0000404e+00	1.0000404e+00	1.0001015e+00	1.0001014e+00
1.2565314e-01	1.0000215e+00	1.0000215e+00	1.0000214e+00	1.0000216e+00
6.9769392e-02	1.0000491e+00	1.0000491e+00	1.0001563e+00	1.0001559e+00
-1.1772370e-01	1.0000190e+00	1.0000190e+00	1.0001191e+00	1.0001191e+00
-1.2566354e-01	1.0000397e+00	1.0000397e+00	1.0000038e+00	1.0000042e+00
-1.0086616e-01	1.0000071e+00	1.0000071e+00	1.0003370e+00	1.0003370e+00
-2.6881501e-01	1.0000454e+00	1.0000454e+00	1.0000034e+00	1.0000033e+00
-7.0281562e-02	*1.0000340e+00*	*1.0000340e+00*	*1.0001240e+00*	*1.0001240e+00*

$$\mathrm{rms}\left(\mathbf{V}_{\mathrm{rotComparison}}\right) = \sqrt{\frac{\|\mathbf{V}_{\mathrm{votComparison}}\|}{\mathrm{rows}\left(\mathbf{V}_{\mathrm{rotComparison}}\right)\mathrm{cols}\left(\mathbf{V}_{\mathrm{rotComparison}}\right)}} \tag{17}$$

In the case when $\mathbf{V}_{\mathrm{rotDICOM}}$ is exactly equal to $\mathbf{V}_{\mathrm{rot}}$ this rms value will be $\frac{1}{3}$. Similarly, for the center of k-space vector, we can define an analogous set of operations. Since the comparison vector in the k-space case would in the ideal case be $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$, the rms value would in this case be exactly 1.

Table 1 shows the results of the above comparisons, taken across 10 sample cases. The first column contains the particular in plane rotation radian. The second and third columns are $\mathrm{rms}\left(\mathbf{V}_{\mathrm{rotComparison}}\right)$ for the direct parsing and indirect calculation cases respectively. For ease of reading, these columns are multiplied by 3 so as to "normalise" the rms. A value of 1, therefore, indicates an exact match between the DICOM rotational `vox2ras` and the methods of this paper. The final two columns are the rms values for the center-of-k-space, with column four corresponding to the direct parsing approach, and column five the indirect calculation method (note that these methods only refer to determining the direction cosines – the k-space center is a linear function of these direction cosines as per Equation 14).

4 Conclusion

This technical report documents several techiques for determining the `vox2ras` matrix for an off-line scan given only scanner raw data. The first, most accurate, method relied on embedding a Siemens-specific rotation matrix at the end of a sequence's `meas.asc` header file. This matrix is manipulated into the rotational components of a `vox2ras` matrix with vector transpose, scaling, and sign operators. While appealing, this method relies on non-standard enhancements to specific scan sequences and is therefore not generally available.

A second method uses the standard `meas.asc` file and applies some knowledge of how Siemens constructs reference vectors to determine the rotational components of a `vox2ras` matrix. This method is portable (i.e. does not rely on non-standard enhancements) and results in rotational components that compare very favourably (if not exactly) with those of direct parsing.

Once the rotational components have been found by either method, determining the center of k-space follows in a straightforward manner.

The accuracy of these methods, as compared to a canonical reference, is very high. In the case of the rotational components, accuracy to at least four decimal places is measured (and this is probably due the fact that the canonical DICOM reference only supplied accuracy to four decimal places). The center of k-space component of the `vox2ras` matrix also compares favorably to the DICOM reference – accurate to about three decimal places.

A working prototype in Matlab of these methods is available in [3].

References

[1] HEMA. Digital Imaging and Communications in Medicine. http://medical.nema.org/, 2004.

[2] NMR Center developed software. tkmedit. NMR Center developed software, 2004.

[3] Rudolph Pienaar. vox2ras_dfmeas.m. NMR Center developed software, 2004.

[4] Rudolph Pienaar. vox2ras_ksolve.m. NMR Center developed software, 2004.

[5] Rudolph Pienaar. vox2ras_rsolveaa.m. NMR Center developed software, 2004.

[6] Rudolph Pienaar. vox2ras_rsolve.m. NMR Center developed software, 2004.

[7] Siemens AG. *IDEA Manual.* Siemens AG, Medical Solutions, Magnetic Resonance, 2003.

[8] Andre van der Kouwe. autoaligncorrect.cpp. NMR Center developed software, 2002.

[9] Harald Werthner. *ICE User's Guide.* Siemens AG, Medical Solutions, Magnetic Resonance, 2002.

A Appendix: second method of indirect rotational `vox2ras` calculation

The primary indirect method of calculating the rotational `vox2ras` component is described in Section 2.4. This relies on a knowledge of the Siemens mechanism of calculating the reference positions for the PhaseEncode and ReadOut vectors. In the absence of this reference knowledge, the method described in this appendix can also be used.

Before we begin, we should note that in the case when there is no relative orientation of the scanned object,

$$\mathbf{R}^T = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{18}$$

the `vox2ras` rotational component $\mathbf{V}_{\text{rot}}$ is found to be (from Equations 3 and 8)

$$\mathbf{V}_{\text{rot}} = \begin{bmatrix} 0 & 0 & -1.33 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{19}$$

Referring to Equation 2, we note that the `meas.asc` file contains the normalized sagittal direction cosine, $\mathbf{z}$ in the fields described in Algorithm 3. We can scale this direction vector by the voxel size value d_z and propose a candidate `vox2ras` rotational component matrix of the following structure

$$\mathbf{V}_{\text{rotCandidate}} = \begin{bmatrix} x_r & y_r & z_r \\ -1 & y_a & z_a \\ x_s & -1 & z_s \end{bmatrix} \tag{20}$$

Since we know that all three column vectors are orthogonal, their dot products are zero. Furthermore, we also know that $\mathbf{z} = -\mathbf{x} \times \mathbf{y}$ (the negation is required to orientate the sagittal direction with the "nose" to the "right" when viewed in `tkmedit`). Given this information, we can construct the following sets of Equations. First, the dot product of $\mathbf{x}$ and $\mathbf{y}$ gives:

$$x_r y_r - y_a - x_s = 0 \tag{21}$$

and the dot product of $\mathbf{x}$ and $\mathbf{z}$ becomes

$$\begin{aligned} x_r z_r - z_a + x_s z_s &= 0 \\ x_r &= \frac{1}{z_r}(z_a - x_s z_s) \end{aligned} \tag{22}$$

while the dot product of $\mathbf{y}$ and $\mathbf{z}$ is

$$\begin{aligned} y_r z_r + y_a z_a - z_s &= 0 \\ y_r &= \frac{1}{z_r}(z_s - y_a z_a) \end{aligned} \tag{23}$$

and finally, using the cross product relation, we can state that

$$\begin{aligned} z_r &= y_a x_s - 1 \\ y_a &= \frac{1}{x_s}(z_r + 1) \end{aligned} \tag{24}$$

Using these four equations we can solve for the unknown components x_r, x_s, y_r and y_a, eventually deriving the following quadratic equation in x_s

$$\begin{array}{ccc} \left(z_r^2 + z_s^2\right) x_s^2 & - \left(2 z_a z_s + z_r z_a z_s\right) x_s & + \left(z_r z_a^2 + z_a^2 + z_r^3 + z_r^2\right) = 0 \\ A x_x^2 & + B x_s & + C = 0 \end{array} \tag{25}$$

which can be solved in a straightforward manner using the well known quadratic relation

$$x_s = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{26}$$

Due to the quadratic nature of the solution, we can have *two* candidate `vox2ras` rotational matrices. Moreover, the `meas.asc` file defines an in-plane rotational parameter, `sSliceArray.asSlice[0].dInPlaneRot` that is relative to a base reference direction (see Section 2.4).

In the absence of knowing how Siemens defines this reference orientation, we can reason as follows: by fixing the two components of the $\mathbf{x}$ and $\mathbf{y}$ matrix as in Equation 20, our solution candidates define an $\mathbf{x}$ and $\mathbf{y}$ vector whose resultant is orientated along a default fixed direction. We still need to determine the relation between this fixed direction and the `dInPlaneRot` parameter of `meas.asc`, θ_{meas} such that $\theta_f = \mathcal{F}(\theta_{\text{meas}})$. Once we know θ_f we can rotate $\mathbf{x}$ and $\mathbf{y}$ about $\mathbf{z}$ by θ_f to determine the rotational `vox2ras` using

$$\mathbf{V}_{\text{rot}} = \mathbf{V}_{\text{rotCandidate}} \begin{bmatrix} \cos\theta_f & \sin\theta_f & 0 \\ -\sin\theta_f & \cos\theta_f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{27}$$

By always using the positive square root in Equation 26, we can empirically study existing `dInPlaneRot` values θ_{meas} as they correspond to `vox2ras` matrices by iteratively solving for θ_f until $\mathbf{V}_{\text{rot}}$ for a given study corresponds to a DICOM-derived rotational `vox2ras`. After analyzing ten different studies, the following linear relationship between θ_{meas} and θ_f was found

$$\theta_f = m\theta_{\text{meas}} + b \tag{28}$$

where $m \cong -1.0025,\ b \cong -0.5188$.

Due to the empirical nature of this analysis, the resultant $\mathbf{V}_{\text{rot}}$ is not as accurate as that found by using the methods of Section 2.4. We can achieve a good correlation between these two methods *a posteriori* by asking ourselves which rotation angle θ_c in Equation 27 would rotate $\mathbf{V}_{\text{rotCandidate}}$ so that it is coincident with the Siemens reference. If we replace $\mathbf{V}_{\text{rotCandidate}}$ with the rotational component of Equation 1, and implement a θ_c rotation, we find a reference matrix

$$\begin{aligned}
\mathbf{V}_{\text{ref}} &= \begin{bmatrix} x_r & y_r & z_r \\ -1 & y_a & z_a \\ x_s & -1 & z_s \end{bmatrix} \begin{bmatrix} \cos\theta_c & \sin\theta_c & 0 \\ -\sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} x_r\cos\theta_c - y_r\sin\theta_c & x_r\sin\theta_c + y_r\cos\theta_c & z_r \\ -\cos\theta_c - y_a\sin\theta_c & -\sin\theta_c + y_a\cos\theta_c & z_a \\ x_s\cos\theta_c + \sin\theta_c & x_s\sin\theta_c - \cos\theta_c & z_s \end{bmatrix} \qquad (29)
\end{aligned}$$

Since from Section 2.4 we know that the last element of the `PhaseEncode` vector (i.e. the last element of the first column) is zero, we can solve for θ_c

$$\begin{aligned}
x_s\cos\theta_c + \sin\theta_c &= 0 \\
\sin\theta_c &= -x_s\cos\theta_c \\
\theta_c &= \arctan -x_s \qquad (30)
\end{aligned}$$

Therefore, in Equation 27, instead of using Equation 28 for θ_f, we can propose

$$\theta_f = \theta_c - \theta_{\text{meas}} \qquad (31)$$

and arrive at a $\mathbf{V}_{\text{rot}}$ that is identical to that found by the methods of Section 2.4. The method described in this appendix is prototyped in [6].