Real-Time 3D Image Registration for Functional MRI

Robert W. Cox and Andrzej Jesmanowicz

Biophysics Research Institute
Medical College of Wisconsin
8701 Watertown Plank Road
Milwaukee, WI 53226

This manuscript is a preprint of a paper that appeared in *Magnetic Resonance in Medicine*, **42**: 1014–1018, 1999.

Abstract

Subject head movements are one of the main practical difficulties with brain functional MRI. A fast accurate method for rotating and shifting a 3D image using a shear factorization of the rotation matrix is described. Combined with gradient descent (repeated linearization) on a least squares objective function, 3D image realignment for small movements can be computed as rapidly as whole brain images can be acquired on current scanners.

INTRODUCTION

Subject head movements are a major problem with brain functional MRI. If two neighboring voxels differ in intrinsic brightness by 20%, then a motion of 10% of a voxel dimension can result in a 2% signal change—comparable to the BOLD signal change at 1.5 Tesla, subsequent to neural activation (1). If the movements occur synchronously with the task/stimulus alternation, false activations may be detected (2). If the movements are uncorrelated with the task/stimulus alternation, the motion-induced signal changes can interfere with the detection of neurally-induced signal changes, reducing the detected volume of activations.

The most common method for dealing this problem is image registration—realignment of each image back to the orientation and location in which it "ought" to have been acquired. Several methods are in use for this purpose (3, 4, 5, 6). One major complaint about these techniques is that their existing implementations are slow, so that the image realignment post-processing step takes much longer than the data acquisition. A basic component of any registration algorithm is the image rotation and shifting method, since this operation must be carried out several times in the search for the best movement parameters. In this paper, we develop a fast technique for 3D image rotation and shifting. Combined with gradient descent on a least squares objective function, we demonstrate that realignment of a 3D image time series can be computed on inexpensive computers as rapidly as whole brain images can be acquired on current scanners.

METHODS

Eddy *et al.* proposed the combination of 3 2D shearing operations and Fourier transform-based shifting for accurate high speed 2D MR image rotation (6). The basis for this method is the following factorization of a general 2D (planar) rotation matrix (7, 8, 9):

$$\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} = \begin{bmatrix} 1 & -\tan\frac{1}{2}\phi \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\phi & 1 \end{bmatrix} \begin{bmatrix} 1 & -\tan\frac{1}{2}\phi \\ 0 & 1 \end{bmatrix} . \qquad [1]$$

They also pointed out that an arbitrary 3D proper orthogonal matrix can be factored into three 2D rotations, so that a general 3D image rotation could be accomplished with nine 2D shears.

Our technique is a generalization of the shear factorization concept directly to 3 dimensions. We define the three basic 3D shear matrices:

$$\mathbf{S}_1(\alpha,\beta)=\begin{bmatrix}1&\alpha&\beta\\0&1&0\\0&0&1\end{bmatrix}\quad \mathbf{S}_2(\alpha,\beta)=\begin{bmatrix}1&0&0\\\alpha&1&\beta\\0&0&1\end{bmatrix}\quad \mathbf{S}_3(\alpha,\beta)=\begin{bmatrix}1&0&0\\0&1&0\\\alpha&\beta&1\end{bmatrix}.$$

In the Appendix we show that any proper orthogonal matrix $\mathbf{A}$ that is not a 180° rotation about a coordinate axis can be factored into the product of four such 3D shear matrices. (A 2D 180° rotation also cannot be factored into three 2D shears; however, a 180° image rotation about a coordinate axis can be carried out simply by re-shuffling elements of the image array.)

The use of this 3D shear factorization provides the same advantage that 2D shears do: the elementary operations are coordinate shifts on 1D rows extracted from the image. Each 1D row can be processed separately; there is no "crosstalk" between the rows when applying such a shear to an image. For example, applying the x-shear matrix $\mathbf{S}_1(\alpha_1,\beta_1)$ and the x-translation δ_1 to $\mathbf{x}_{\text{old}}$ yields

$$\begin{bmatrix}x_{\text{new}}\\y_{\text{new}}\\z_{\text{new}}\end{bmatrix}=\begin{bmatrix}x_{\text{old}}+\alpha_1 y_{\text{old}}+\beta_1 z_{\text{old}}+\delta_1\\y_{\text{old}}\\z_{\text{old}}\end{bmatrix}.$$

Any particular x-row of a 3D image (with $y=y_{\text{old}}=y_{\text{new}}$ and $z=z_{\text{old}}=z_{\text{new}}$ both fixed) is shifted by the fixed amount $\alpha_1 y+\beta_1 z+\delta_1$, which requires interpolating the row's data to the new shifted grid.

A highly accurate and reasonably efficient way to do this shifting interpolation is with 2 1D FFTs, applying a linear-in-frequency phase shift in the frequency domain. This method for image rotation is the full 3D generalization of the method of Eddy *et al.* (6). Alternatives to Fourier resampling along each row are polynomial methods; for example, 3$^{\text{rd}}$, 5$^{\text{th}}$, and 7$^{\text{th}}$ order Lagrange polynomial interpolation are very similar to Hamming tapered sinc interpolation over $\pm 2\pi$, $\pm 3\pi$, and $\pm 4\pi$, respectively. These methods are considerably faster than the Fourier method, and the 7$^{\text{th}}$ order (heptic) interpolation method is nearly as accurate for many purposes.

Accurate use of the rotation algorithm requires choosing the axes ordering that results in the least intermediate image distortion, say as measured by $\max\{|\alpha_i|,|\beta_i|\}$. In our implementation, the shear factorizations are computed for all 6 orderings, and the ordering with the smallest distortion is selected. In addition, rotations of nearly 180° require very large intermediate shearing. Since it is easy to flip a 3D image by 180° about x, y, or z, choosing the proper flip will make the algorithm more stable for large rotations, by converting the rotation back to a smaller angle. In our implementation, the flip is chosen to minimize the net rotation angle $2\cos^{-1}[\frac{1}{2}(1+A_{11}+A_{22}+A_{33})^{1/2}]$, prior to the shear factorization. In the registration algorithm outlined below, large angles and large intermediate shearings are not encountered, but the image rotation software allows for them for the sake of completeness. The computational overhead is negligible compared to the many interpolations required.

Functional MRI generally requires only small rotations and translations: 1–2 degrees and 1–2 voxel dimensions. These small effects mean that registration of a base image $J(\mathbf{x})$ to a target image $I(\mathbf{x})$ can be accomplished rapidly by repeated linearization of the weighted least squares penalty function

$$E(\mathbf{a}) = \sum_{\mathbf{x}} w(\mathbf{x}) \left[J(T[\mathbf{a}]\mathbf{x}) - I(\mathbf{x})\right]^2,$$

with respect to the motion parameters $\mathbf{a} \in \mathbf{R}^6$, comprising 3 rotation angles and 3 translations (4, 10). Here, $w(\mathbf{x})$ is a non-negative weighting function and $T[\mathbf{a}]$ is the spatial transformation corresponding to $\mathbf{a}$. Once the best-fit $\mathbf{a}$ is determined, the registered image is $I(T[\mathbf{a}]^{-1}\mathbf{x})$. (Repeated linearization is equivalent to applying a gradient-descent algorithm to $E(\mathbf{a})$.)

In our implementation, we choose $w(\mathbf{x})$ to be a smoothed version of the base image $J(\mathbf{x})$. We have also experimented with choosing $w(\mathbf{x})$ to be inversely proportional to the image time series variance at $\mathbf{x}$ after an initial registration pass through the data (this weighting cannot be used in real-time processing). This latter weighting is intended to minimize the influence of regions with large intrinsic variability, such as the brainstem and functionally active areas. Significant improvement in the detection of activated voxels has not been observed with this type of weighting, but investigations along this line are continuing.

Table 1. CPU time (s) for one Fourier-resampled 3D rotation.

	128×128×30	256×256×124
SGI R10000 175 MHz	1.5	27.0
HP PA–8000 200 MHz	1.3	24.7
Pentium II 400 MHz	0.8	13.8

Table 2. CPU time (s) for one heptic-resampled 3D rotation.

	128×128×30	256×256×124
SGI R10000 175 MHz	1.0	15.8
HP PA–8000 200 MHz	0.6	10.7
Pentium II 400 MHz	0.5	7.3

Table 3. CPU time (s) for 3D registration of a time series comprising 80 128×128×30 images Acquisition time was 264 s.

	Fourier	heptic	quintic	cubic
SGI R10000 175 MHz	335.0	227.5	213.3	204.7
HP PA–8000 200 MHz	276.2	127.6	115.6	107.0
Pentium II 400 MHz	162.1	100.7	90.0	83.6

RESULTS

Table 1 shows the CPU time needed for a single 3D image rotation using Fourier-based resampling, executed on three types of Unix workstations. Table 2 shows the CPU times using the heptic resampling method, which is considerably faster. We have adopted this method as our default interpolation scheme. The "raw" speeds of these CPUs, as measured in peak Mflops attainable with a 1D FFT routine, are similar. The differences in timings seem to be more related to compilers, cache, and memory access.

We acquired several MR image time series while the subject was instructed to move his slowly at will. The scanner used was a Bruker Biospec 3 Tesla, using single-shot full k-space EPI on a 96×96 matrix, reconstructed to 128×128 images, with a 166 KHz bandwidth and a 240 mm field-of-view. Thirty contiguous 4 mm thick sagittal slices were gathered covering most of the cerebrum, with TE=41.6 ms and TR=3300 ms (110 ms per slice). Eighty volumes were gathered in each imaging run (264 s total).

Table 3 shows the average CPU times needed for registration of these EPI time series to a base image. These CPU times include initialization of the registration algorithm, including computation of $\nabla_{\mathbf{a}}J$ by finite differences and Cholesky factorization of the normal equations for the linearized least squares solution. The motion estimates from this algorithm were essentially the same—within 0.05° and 0.04 mm—as those of *AIR* 3.08 (5), a widely used and tested software package. Our new algorithm, using its default heptic interpolation, executed an average of 4.3 times faster than *AIR*, using its lower accuracy default trilinear interpolation. The ranges of movement parameters were $\pm 2^\circ$ and ± 2 mm. When viewed in cine mode, the subject's movements were quite obvious before registration, and were much less noticeable after registration.

DISCUSSION

Heptic interpolation is faster than the data acquisition on all systems tested, showing that accurate real-time 3D registration is achievable on commonly available computers. On a Pentium II, the implementation using Fourier interpolation is capable of keeping up with real-time. Slightly faster CPUs could perform real-time registration with Fourier interpolation on complex-valued images, which would preserve phase information and keep the full bandwidth of the MRI data without aliasing (8).

The registration method described herein is now part of the real-time FMRI acquisition and activation analysis modules within the *AFNI* package (11, 12). In this venue, registration is performed after each 3D volume is complete, and the estimated movement parameters are displayed in a continually updating graph. The real-time image reconstruction software is a separate program, and communicates with *AFNI* using shared memory or TCP/IP sockets for intra- or inter-computer communications, respectively. Such modularity is designed to make it straightforward to use the real-time registration and functional detection system with an entirely different real-time image acquisition system.

The display of estimated subject head motion during FMRI scanning sessions has proved to be a very useful quality assurance tool. If the estimated movements are too large, it is possible to reacquire the functional image time series immediately. For many subjects, we have also found that feedback from the investigator or scanner operator will reduce the amount of motion significantly. Most subjects simply don't have a good proprioceptive "feel" for how still they must remain in the magnet. A little verbal feedback can provide the training needed to gain this knowledge within a few minutes.

The rotation and registration algorithms are also included in command line (batch) programs for offline analyses, and in a plugin for the interactive *AFNI* analysis and visualization program. This software is freely available from the authors; information can be found at `http://www.biophysics.mcw.edu`.

APPENDIX

We show that for any proper orthogonal $\mathbf{A}$, there are four 3D shears so that

$$[\mathbf{S}_i^{(3)}]^{-1}[\mathbf{S}_k^{(2)}]^{-1}[\mathbf{S}_j^{(1)}]^{-1}[\mathbf{S}_i^{(0)}]^{-1}\mathbf{A} = \mathbf{I} \quad \text{or} \quad \mathbf{A} = \mathbf{S}_i^{(0)}\mathbf{S}_j^{(1)}\mathbf{S}_k^{(2)}\mathbf{S}_i^{(3)} \ ,$$

where the shear axes ordering (i, j, k) is some permutation of $(1, 2, 3)$.

We denote the first transformed matrix $([\mathbf{S}_i^{(0)}]^{-1}\mathbf{A})$ by $\mathbf{A}^{(1)}$, the second $([\mathbf{S}_j^{(1)}]^{-1}[\mathbf{S}_i^{(0)}]^{-1}\mathbf{A})$ by $\mathbf{A}^{(2)}$, etc. It is convenient to start the analysis with $\mathbf{A}^{(1)}$ and determine the subsequent shear transformations, only coming back to calculate the first shear at the end. To begin with, we will assume that the axes ordering $(i, j, k) = (3, 1, 2)$, so that the second shear $\mathbf{S}_i^{(1)}$ is an x-shear:

$$\mathbf{A}^{(2)} \equiv [\mathbf{S}_1^{(1)}]^{-1}\mathbf{A}^{(1)} = \begin{bmatrix} 1 & -\alpha^{(1)} & -\beta^{(1)} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1^{(1)} & a_2^{(1)} & a_3^{(1)} \\ b_1^{(1)} & b_2^{(1)} & b_3^{(1)} \\ c_1^{(1)} & c_2^{(1)} & c_3^{(1)} \end{bmatrix}$$

$$= \begin{bmatrix} a_1^{(1)} - b_1^{(1)}\alpha^{(1)} - c_1^{(1)}\beta^{(1)} & a_2^{(1)} - b_2^{(1)}\alpha^{(1)} - c_2^{(1)}\beta^{(1)} & a_3^{(1)} - b_3^{(1)}\alpha^{(1)} - c_3^{(1)}\beta^{(1)} \\ b_1^{(1)} & b_2^{(1)} & b_3^{(1)} \\ c_1^{(1)} & c_2^{(1)} & c_3^{(1)} \end{bmatrix}$$

We wish to choose the shear parameters $(\alpha^{(1)}, \beta^{(1)})$ so that the first row of $\mathbf{A}^{(2)}$ is $[1\ \ 0\ \ 0]$. Making the last two elements of this row equal to zero requires

$$\begin{bmatrix} b_2^{(1)} & c_2^{(1)} \\ b_3^{(1)} & c_3^{(1)} \end{bmatrix} \begin{bmatrix} \alpha^{(1)} \\ \beta^{(1)} \end{bmatrix} = \begin{bmatrix} a_2^{(1)} \\ a_3^{(1)} \end{bmatrix} \ . \qquad [2]$$

We denote the 2×2 coefficient matrix in Eq. [2] by $[\mathbf{A}_{(11)}^{(1)}]^T$, since it is formed by striking out the first row and first column of $\mathbf{A}^{(1)}$ and transposing the result. Then $(\alpha^{(1)}, \beta^{(1)})$ are given by

$$\begin{bmatrix} \alpha^{(1)} \\ \beta^{(1)} \end{bmatrix} = \left[[\mathbf{A}_{(11)}^{(1)}]^T\right]^{-1} \begin{bmatrix} a_2^{(1)} \\ a_3^{(1)} \end{bmatrix} \ .$$

With these shear parameters, the first element of the first row of $\mathbf{A}^{(2)}$ becomes

$$a_1^{(2)} = a_1^{(1)} - [b_1^{(1)}\ c_1^{(1)}] \left[[\mathbf{A}_{(11)}^{(1)}]^T\right]^{-1} \begin{bmatrix} a_2^{(1)} \\ a_3^{(1)} \end{bmatrix} = \frac{\det[\mathbf{A}^{(1)}]}{\det[\mathbf{A}_{(11)}^{(1)}]} \ .$$

Since $\det[\mathbf{A}^{(1)}] = 1$, the first row of $\mathbf{A}^{(2)}$ can be transformed to $[1\ 0\ 0]$ by an x-shear if and only if $\det[\mathbf{A}_{(11)}^{(1)}] = b_2^{(1)}c_3^{(1)} - b_3^{(1)}c_2^{(1)} = 1$ (which also guarantees that $[\mathbf{A}_{(11)}^{(1)}]^T$ is nonsingular).

Assuming that this condition is met, the next stage in the factorization is to apply a y-shear $[\mathbf{S}_2^{(2)}]^{-1}$ to $\mathbf{A}^{(2)}$ to make the second row of the transformed matrix $\mathbf{A}^{(3)}$ equal to $[0\ 1\ 0]$. Reasoning similar to that above shows this can be done if and only if $\det[\mathbf{A}_{(22)}^{(2)}] = c_3^{(1)} = 1$.

The final stage is to apply a z-shear $[\mathbf{S}_3^{(3)}]^{-1}$ to $\mathbf{A}^{(3)}$ to make the third row of $\mathbf{A}^{(4)}$ equal to $[0\ \ 0\ \ 1]$, which will make the entire matrix $\mathbf{A}^{(4)}$ equal to the identity matrix $\mathbf{I}$. If the transformation to $\mathbf{A}^{(3)}$ is possible, this final stage is also possible, with $\alpha^{(3)} = c_2^{(1)}$ and $\beta^{(3)} = c_3^{(1)} = 1$.

Thus, any matrix $\mathbf{A}^{(1)}$ with $\det[\mathbf{A}^{(1)}] = 1$ can be factored as a product of three shears $\mathbf{S}_1^{(1)}\mathbf{S}_2^{(2)}\mathbf{S}_3^{(3)}$ if the elements of $\mathbf{A}^{(1)}$ satisfy the two conditions below:

$$b_2^{(1)} - b_3^{(1)} c_2^{(1)} = 1 \qquad \text{and} \qquad c_3^{(1)} = 1 \; . \tag{[3]}$$

These conditions are clearly not true for general 3D rotation matrices, but they can be forced to be true by applying an initial z-shear to the rotation matrix $\mathbf{A}$:

$$\mathbf{A}^{(1)} \equiv [\mathbf{S}_3^{(0)}]^{-1}\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\alpha^{(0)} & -\beta^{(0)} & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$$

$$= \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 - a_1\alpha^{(0)} - b_1\beta^{(0)} & c_2 - a_2\alpha^{(0)} - b_2\beta^{(0)} & c_3 - a_3\alpha^{(0)} - b_3\beta^{(0)} \end{bmatrix} .$$

The motivation for using $\mathbf{S}_3$ here is that both conditions in Eq. [3] involve the third row of $\mathbf{A}^{(1)}$, and these elements can be modified only by a z-shear. Requiring $c_3^{(1)} = 1$ and $c_2^{(1)} = (b_2^{(1)} - 1)/b_3^{(1)}$ yields two equations for the shear parameters $(\alpha^{(0)}, \beta^{(0)})$:

$$\begin{bmatrix} a_2 & b_2 \\ a_3 & b_3 \end{bmatrix} \begin{bmatrix} \alpha^{(0)} \\ \beta^{(0)} \end{bmatrix} = \begin{bmatrix} c_2 - (b_2^{(1)} - 1)/b_3^{(1)} \\ c_3 - 1 \end{bmatrix} . \tag{[4]}$$

Equation [4] will have a solution if $b_3 \neq 0$ and $a_2b_3 - b_2a_3 \neq 0$. If both these conditions hold, then $\mathbf{A}$ can be transformed to $\mathbf{A}^{(1)}$ where Eq. [3] holds. Under these conditions, $\mathbf{A}$ can thus be factored into four shears $\mathbf{S}_3^{(0)}\mathbf{S}_1^{(1)}\mathbf{S}_2^{(2)}\mathbf{S}_3^{(3)}$.

In a general rotation matrix $\mathbf{A}$, it might not be true that $a_2b_3 - b_2a_3 \neq 0$. This is the determinant of the submatrix $\mathbf{A}_{(31)}$ (striking out the third row and first column of $\mathbf{A}$), which arises because the order of application of the first two shears in the reduction of $\mathbf{A}$ to $\mathbf{I}$ was to the third and first rows ($\mathbf{S}_3^{(0)}$ and $\mathbf{S}_1^{(1)}$), respectively. It is possible to attempt the shear factorization using different axes orderings: if (i, j, k) is one of the six permutations of $(1, 2, 3)$, we can try to transform $\mathbf{A}$ to the identity matrix by applying shears $[\mathbf{S}_i^{(3)}]^{-1}[\mathbf{S}_k^{(2)}]^{-1}[\mathbf{S}_j^{(1)}]^{-1}[\mathbf{S}_i^{(0)}]^{-1}$. This transformation is possible if $\det[\mathbf{A}_{(ij)}] \neq 0$ and matrix element $A_{ki} \neq 0$.

Any 3D matrix $\mathbf{A}$ with $\det[\mathbf{A}] = 1$ and that also has $\det[\mathbf{A}_{(ij)}] = 0$ for all $i \neq j$ will not be able to be factored into four shears using the technique above. To find all matrices that satisfy these seven conditions, we used the Gröbner basis package in the computer algebra software *Maple V* (Waterloo Maple, Inc., Waterloo, Ontario) to solve the set of polynomial equations. There is a single two-parameter family of such matrices:

$$\mathbf{A} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \nu & 0 \\ 0 & 0 & (\mu\nu)^{-1} \end{bmatrix} .$$

For such an $\mathbf{A}$ to be orthogonal, it must have $|\mu| = |\nu| = 1$. If both $\mu = \nu = 1$, then $\mathbf{A} = \mathbf{I}$, and no factorization is needed. Otherwise, exactly two of the diagonal elements of $\mathbf{A}$ are -1

and one diagonal element is 1. Such a matrix represents a 180° rotation about the coordinate axis corresponding to the diagonal element that equals 1.

For a given axes ordering (i, j, k) *not* to have a shear factorization computable by our technique, one of $\det[\mathbf{A}_{(ij)}] = 0$ or $A_{ki} = 0$ must be true. If $\mathbf{A}$ is such that no shear factorization exists for *any* ordering, then for each of the six possible orderings one of these two conditions must hold. There are $2^6 = 64$ combinations of conditions that would prevent $\mathbf{A}$ from being factored into four shears in all six orderings. The previous paragraph analyzed the case where the conditions $\det[\mathbf{A}_{(ij)}] = 0$ held for all orderings. The opposite case has $A_{ki} = 0$ for all orderings, in which case $\mathbf{A}$ must be diagonal, and the same conclusion follows. One example of a mixed set of equations would be to require $\det[\mathbf{A}_{(ij)}] = 0$ for orderings $(i, j, k) = (1, 2, 3)$, $(1, 3, 2)$, and $(3, 1, 2)$, and to require $A_{ki} = 0$ for the other three permutations. (Solving this example will result in Eq. [5], below.)

All 64 cases were analyzed using *Maple V*. For each case, six equations that would block the shear factorization of $\mathbf{A}$ were selected. Seven additional equations in the system were $\det[\mathbf{A}] = 1$, and the requirement that the rows and columns of $\mathbf{A}$ be pairwise orthogonal. Only three classes of orthogonal solutions exist: the identity matrix, 180° rotations about a coordinate axis, or matrices that are actually 2D rotations, one example of which is

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} , \qquad [5]$$

where ϕ is an arbitrary parameter (of course, the identity matrix and 180° rotations about coordinate axes are special cases of 2D rotations). If $\phi \neq \pi$, such a 2D rotation matrix can be factored into three 2D shears (*cf.* Eq. [1]), so this class of solutions does not in fact provide any new examples of matrices that cannot be factored into four 3D shears. It just shows that Eq. [4] can have a singular coefficient matrix but still have a solution. In this case, Eq. [4] becomes

$$\begin{bmatrix} 0 & \cos\phi \\ 0 & \sin\phi \end{bmatrix} \begin{bmatrix} \alpha^{(0)} \\ \beta^{(0)} \end{bmatrix} = \begin{bmatrix} -\sin\phi - (\cos\phi - 1)/\sin\phi \\ \cos\phi - 1 \end{bmatrix} ,$$

which has the solution $\alpha^{(0)} = 0$, $\beta^{(0)} = (\cos\phi - 1)/\sin\phi$. Thus we have shown that all 3D rotation matrices that are not 180° rotations about a coordinate axis can be factored into four 3D shears, using at least one axes ordering.

Allowing for a translation as well as a rotation is most easily analyzed by increasing the matrix and vector dimensions to four:

$$\tilde{\mathbf{A}} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{s} \\ \hline 0\,0\,0 & 1 \end{array}\right] \qquad \tilde{\mathbf{x}} = \left[\begin{array}{c} \mathbf{x} \\ \hline 1 \end{array}\right] \qquad \tilde{\mathbf{x}}_{\text{new}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_{\text{old}} = \left[\begin{array}{c} \mathbf{A}\mathbf{x}_{\text{old}} + \mathbf{s} \\ \hline 1 \end{array}\right] ,$$

so that the augmented transformation takes $\mathbf{x}_{\text{old}} = [x_{\text{old}}\ y_{\text{old}}\ z_{\text{old}}]^T$ to $\mathbf{x}_{\text{new}} = \mathbf{A}\mathbf{x}_{\text{old}} + \mathbf{s}$, where $\mathbf{s} = [s_1\ s_2\ s_3]^T$ is the desired translation vector. In this notation, the $(i, j, k) = (3, 1, 2)$ factorization of $\tilde{\mathbf{A}}$ becomes

$$\tilde{\mathbf{A}} = \left[\begin{array}{c|c} & 0 \\ \mathbf{S}_3^{(0)} & 0 \\ & \delta_3 \\ \hline 0\,0\,0 & 1 \end{array}\right] \left[\begin{array}{c|c} & \delta_1 \\ \mathbf{S}_1^{(1)} & 0 \\ & 0 \\ \hline 0\,0\,0 & 1 \end{array}\right] \left[\begin{array}{c|c} & 0 \\ \mathbf{S}_2^{(2)} & \delta_2 \\ & 0 \\ \hline 0\,0\,0 & 1 \end{array}\right] \left[\begin{array}{c|c} & 0 \\ \mathbf{S}_3^{(3)} & 0 \\ & 0 \\ \hline 0\,0\,0 & 1 \end{array}\right] , \qquad [6]$$

where the parameters $(\delta_1, \delta_2, \delta_3)$ are chosen to match the desired translation vector $\mathbf{s}$. Multiplying out Eq. [6], we find

$$\mathbf{s} = \mathbf{S}_3^{(0)}\mathbf{S}_1^{(1)} \begin{bmatrix} 0 \\ \delta_2 \\ 0 \end{bmatrix} + \mathbf{S}_3^{(0)} \begin{bmatrix} \delta_1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \delta_3 \end{bmatrix} = \begin{bmatrix} \delta_1 + \alpha_1\delta_2 \\ \delta_2 \\ \delta_3 + \alpha_0\delta_1 + (\alpha_0\alpha_1 + \beta_0)\delta_2 \end{bmatrix}. \qquad [7]$$

Solving Eq. [7] yields $\delta_1 = s_1 - \alpha_1 s_2$, $\delta_2 = s_2$, and $\delta_3 = s_3 - \alpha_0 s_1 - \beta_0 s_2$.

References

1. Bandettini PA, Wong EC, Hinks RS, Tikofsky RS, and Hyde JS. Time course EPI of human brain function during task activation. Magn Reson Med 1992; **25**: 390–397.

2. Hajnal JV, Myers R, Oatridge A, Schwieso JE, Young IR, and Bydder GM. Artifacts due to stimulus correlated motion in functional imaging of the brain. Magn Reson Med 1994; **31**: 283–291.

3. Hajnal JV, Saeed N, Soar EJ, Oatridge A, Young IR, and Bydder GM. A registration and interpolation procedure for subvoxel matching of serially acquired MR images. J Comput Assist Tomog 1995; **19**: 289–296.

4. Friston KJ, Ashburner J, Frith CD, Poline J-B, Heather JD, and Frackowiak RSJ. Spatial registration and normalization of images. Human Brain Mapping 1995; **3**: 165–189.

5. Woods RP, Grafton ST, Holmes CJ, Cherry SR, and Mazziotta JC. Automated image registration: I. General methods and intrasubject, intramodality validation. J Comput Assist Tomog 1998; **22**: 139–152.

6. Eddy WF, Fitzgerald M, and Noll DC. Improved image registration by using Fourier interpolation. Magn Reson Med 1996; **36**: 923–931.

7. Paeth AW. A fast algorithm for general raster rotation, *in* "Proc. Graphics Interface '86, Canadian Information Processing Society, Vancouver, 1986," pp. 77–81.

8. Fraser D and Schowengerdt RA. Avoidance of additional aliasing in multipass image rotations. IEEE Trans Image Proc 1994; **3**: 721–735.

9. Unser M, Thévenaz P, and Yaroslavsky L. Convolution-based interpolation for fast, high-quality rotation of images. IEEE Trans Image Proc 1995 **4**: 1371–1381.

10. Lucas BD, and Kanade T. An iterative image registration technique with an application to stereo vision, *in* "Proc. Intl. Joint Conf. Artificial Intelligence, Vancouver, 1981," pp. 674–679.

11. Cox RW, Jesmanowicz A, and Hyde JS. Real-time functional magnetic resonance imaging. Magn Reson Med 1995; **33**: 230–236.

12. Cox RW. AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages. Comput Biomed Res 1996; **29**: 162–173.