Appendix F: Automated Image Registration

The following details on AIR 3.03 are adapted from the Automated Image Registration home page (http://bishopw.loni.ucla.edu/AIR3).

F.1 PURPOSE

The Linear Algorithm option in the AIR dialog box uses the align linear program. This is a general linear intramodality and intermodality registration tool (within or across subjects, 2D or 3D). The user can specify any of a variety of models, including rigid-body, affine, or perspective.

The transformation generated by this program can be saved and used to reslice the other data sets to match the specified standard data set through the shadow transform dialog box.

The Warp Algorithm option in the AIR dialog box uses the align_warp program. This is a nonlinear registration tool that can be used within or across subjects and includes implementation of 2D and 3D nonlinear spatial transformation models.

The transformations generated by this program cannot currently be saved.

F.2 PARAMETERS:

F.2.1 Linear Algorithm Model Menu

F.2.1.1 3-D MODELS:

- 6. rigid body 6 parameter model
- 7. global rescale 7 parameter model
- 9. traditional 9 parameter model (std must be on AC-PC line)
- 12. affine 12 parameter model
- 15. perspective 15 parameter model

F.2.1.2 2-D MODELS (CONSTRAINED TO IN-PLANE/NO INTERPOLATION):

- 23. 2-D rigid body 3 parameter model
- 24. 2-D global rescale 4 parameter model
- 25. 2-D affine/fixed determinant 5 parameter model
- 26. 2-D affine 6 parameter model

F.2.2 Warp Algorithm Model Menu

F.2.2.1 3-D MODELS:

- 12. first order linear 12 parameter model

- 30. second order nonlinear 30 parameter model
- 60. third order nonlinear 60 parameter model
- 105. fourth order nonlinear 105 parameter model
- 168. fifth order nonlinear 168 parameter model

F.2.2.2 2-D MODELS (CONSTRAINED TO IN-PLANE/NO INTERPOLATION):

- 6. first order linear 6 parameter model
- 12. second order nonlinear 12 parameter model
- 20. third order nonlinear 20 parameter model
- 30. fourth order nonlinear 30 parameter model
- 42. fifth order nonlinear 42 parameter model

F.2.2.3 STANDARD IMAGE

The name of the image that you want the other image resliced to match.

F.2.2.4 RESLICE IMAGE OR GROUP

The name of the image or group of images that you want to reslice to match the standard image.

F.2.2.5 THRESHOLD STANDARD IMAGE

Defines a minimum voxel value for the standard image. Voxels in the standard image below this value are excluded from analysis when computing the cost function and its derivatives in the forward direction. The value should always be an integer less than the maximum possible voxel value.

F.2.2.6 THRESHOLD RESLICE IMAGE

Defines a minimum voxel value for the reslice image. Voxels in the reslice image below this value are excluded from analysis when computing the cost function and its derivatives in the reverse direction. The value should always be an integer less than the maximum possible voxel value.

F.2.2.7 PARTITIONS IN STANDARD IMAGE

Defines the number of partitions used for segmenting the standard image when computing the standard deviation of the ratio image in the forward direction. If this value is less than 1, no forward direction computation is performed. A value of 256 is typically used for intermodality registration if the standard image is an MRI study. A value of zero is typically used for intermodality registration if the standard image is a PET study. For intramodality registration, the default value of 1 is appropriate. The Registration Type option sets this option to a value that is typically appropriate for the given type of registration. This option only applies to Linear Registrations.

F.2.2.8 PARTITIONS IN RESLICE IMAGE

Defines the number of partitions used for segmenting the reslice image when computing the standard deviation of the ratio image in the reverse direction. If this value is less than 1, no reverse direction computation is performed. A value of 256 is typically used for intermodality registration if the reslice image is an

MRI study. A value of zero is typically used for intermodality registration if the reslice image is a PET study. For intramodality registration, the default value of 1 is appropriate. The Registration Type option sets this option to a value that is typically appropriate for the given type of registration. This option only applies to Linear Registrations.

F.2.2.9 FWHM-X FWHM-Y FWHM-Z (STANDARD IMAGE)

If this option is used, smoothing filters are applied along the x, y and z axes of the standard image before performing registration. The FWHM value specifies the full width at half maximum of the Gaussian smoothing filter to be applied along each dimension. The filters have units of millimeters (or whatever units you use to specify voxel sizes in your images). All three dimensions must be specified. If you give a value of zero, no smoothing will be applied along the corresponding dimension.

F.2.2.10 FWHM-X FWHM-Y FWHM-Z (RESLICE IMAGE)

If this option is used, smoothing filters are applied along the x, y and z axes of the reslice file before performing registration. The FWHM value specifies the full width at half maximum of the Gaussian smoothing filter to be applied along each dimension. The filters have units of millimeters (or whatever units you use to specify voxel sizes in your images). All three dimensions must be specified. If you give a value of zero, no smoothing will be applied along the corresponding dimension.

F.2.2.11 INITIAL SAMPLING

Controls how densely data is sampled during the first iterative cycle of the algorithm. Large values generally speed up the registration process because gross misregistration can be detected with fairly superficial sampling of the data. However, choosing an excessively large value can be counterproductive if the algorithm falls into an infinite loop or is led far from the true value by nonrepresentative sampling. Avoid multiples of two when choosing sampling parameters. If any of your matrix dimensions are divisible by two, the sampling will become spatially biased until the sampling density reaches one, at which point the algorithm will have to iteratively overcome the earlier bias at the maximal sampling density. If your matrix dimensions are divisible by three, you will have a similar problem with sampling densities that are multiples of three.

F.2.2.12 FINAL SAMPLING

Controls how densely data is sampled during the final iterative cycle of the algorithm. If your data is oversampled, the time spent sampling very densely may not provide any significant improvement in accuracy. In such cases, you may want to choose a final_sampling that is greater than one. Iterations will cease if the new sampling density is less than the final_sampling density specified here.

F.2.2.13 SAMPLING DECREMENT RATIO

Determines the number of intermediate iterative cycles of the algorithm. The current sampling is divided by this ratio with each cycle to determine the new sampling.

F.2.2.14 CONVERGENCE THRESHOLD

Controls how small the predicted change in the cost function must be in order to meet the convergence criteria. Setting this value too large will result in convergence while the images are still misregistered; setting it too small may lead to a failure to converge.

F.2.2.15 REPEATED ITERATIONS

Controls the maximum number of iterations permitted at each sampling density. If this number is made too low, it will lead to inaccurate results and/or slow down the overall performance of the algorithm by preventing you from making use of information that could have been derived more quickly at the prematurely aborted, more superficial sampling.

F.2.2.16 HALT-AFTER-(N)-ITERATIONS-WITHOUT-IMPROVEMENT

Controls the maximum number of iterations without any observed improvement in the cost function. If greater than or equal to the "repeated_iterations" variable above, this value has no effect. At lower values, it can help you escape from situations where you are bouncing back and forth between two or three locations in parameter space without making any real progress. This sort of thing usually only happens at superficial sampling densities.

F.2.2.17 ALTERNATE STRATEGY-AFTER-(M)-ITERATIONS-WITHOUT-IMPROVEMENT

Similar to the preceding option except that it does not force termination of the current sampling density, but rather tries to split the difference between the locations in parameter space at the current sampling. If greater than or equal to the "halt-after-(N)-iterations-without-improvement" or the "repeated-n-iterations" variables above, this value has no effect.

F.2.2.18 PRE-ALIGNMENT INTERPOLATION

In contrast to AIR 1.0, this new algorithm in AIR 3.02 does not apply prealignment interpolation of the files to cubic voxels by default. If you want prealignment interpolation, it can be enabled using this flag. Using prealignment interpolation will slow down the algorithm if you have thick slices, but may result in a more robust algorithm. If your voxels are already cubic, prealignment interpolation has no effect.

F.2.2.19 COST FUNCTION

Determines which cost function is used for aligning the images. This should be a number from the corresponding menu:

1. **Standard Deviation of Ratio Images** - This cost function has the advantage of being independent of image intensity, so image intensities can be poorly matched and the registration will not be adversely effected.
2. **Least Squares** - This cost function assumes that the image intensities are scaled identically. Least squares is computationally simpler and therefore faster than the standard deviation of ratio images, but may be inaccurate if the image intensities are poorly matched.

3. **Least Squares with Intensity Rescaling** - This cost function is identical to the least squares cost function except that an intensity scaling term is added to the model.

This option is only available for the Linear Algorithm.

F.2.2.20 DESCRIPTION OF LINEAR ALGORITHM SPATIAL MODELS

3D Models:

6. rigid body 6 parameter model - Used for intra-subject registration when all voxel sizes are known accurately

7. global rescale 7 parameter model - Not too useful for biological data

9. traditional 9 parameter model (standard image must be on AC-PC line) - This is the typical (as opposed to literal) Talairach model, provided that the standard file has been properly oriented using the Talairach rules.

12. affine 12 parameter model - This is the preferred model for intersubject registration

15. perspective 15 parameter model - The perspective distortions are probably not worth the extra computational cost in most cases.

2D Models:

23. 2-D rigid body 3 parameter model - Analogous to the 6 parameter 3D model

24. 2-D global rescale 4 parameter model - Analogous to the 7 parameter 3D model. Might be useful for aligning photos of distant objects taken from various distances.

25. 2-D affine/fixed determinant 5 parameter model - This model allows for nonrigid distortion so long as total area is preserved. This may be a useful model for realigning data from serial tissue sections.

26. 2-D affine 6 parameter model - Analogous to the 12 parameter 3D model.

F.2.2.21 DESCRIPTION OF WARP ALGORITHM SPATIAL MODELS

3D models:

1. first order linear 12 parameter model

2. second order nonlinear 30 parameter model

3. third order nonlinear 60 parameter model

4. fourth order nonlinear 105 parameter model

5. fifth order nonlinear 168 parameter model

2D models:

21. first order linear 6 parameter model

22. second order nonlinear 12 parameter model

23. third order nonlinear 20 parameter model

24. fourth order nonlinear 30 parameter model

25. fifth order nonlinear 42 parameter model

The order specifies the order of the polynomial transformation used as a spatial transformation model. The algorithm will begin with model selected as the Initial Model until convergence and then increment the model order by one until the Final Model is reached.

F.2.2.22 INITIALIZATION FILE (LINEAR)

The name of an ASCII file containing spatial transformation initialization parameters. These parameters can be used to control the starting position for automated registration, a feature that is useful if the initial misregistration is extreme (e.g., >45°; of rotational misregistration) or if the default registration leads to an obviously incorrect result. The format for the rigid-body initialization file is discussed under file types. Rigid-body initialization files are created most easily using the program manualreslice. Different spatial models require different numbers and types of parameters in the initialization file. Note that some cost functions may also allow an intensity parameter initialization file. (Note: MEDx uses the anatomic orientation of each volume to correctly orient the two images prior to calling AIR. Thus, differences due to plane of section need not be accounted for by using this option).

F.2.2.23 INITIALIZATION FILE (WARP)

The name of an ASCII file containing spatial transformation initialization parameters. If you override the default procedure of starting with a first order model, it is very important to use an initialization file (presumably derived from some prior successful registration). Poor initialization of high order transformations can lead to poor results. Each spatial model requires different numbers of parameters in the initialization file. Note that voxel intensity initialization is performed separately in an intensity parameter initialization file. (Note: MEDx uses the anatomic orientation of each volume to correctly orient the two images prior to calling AIR. Thus, differences due to plane of section need not be accounted for by using this option).

F.2.2.24 ASSUME NON-INTERACTION OF SPATIAL PARAMETER DERIVATIVES

Ignoring the second derivatives of the interpolated voxel values with respect to spatial location by selecting this option can significantly improve the speed of the algorithm with little cost in terms of accuracy.

F.3 COMMENTS

F.3.1.1 GENERAL

For MR data, it is recommended that you edit the data to remove nonbrain structures (e.g., scalp, skull and dura). Even if the algorithm does run successfully, you will have invested a lot of computation time in making sure that your subjects' noses are of similar size and shape, even if this means that their cerebellums don't line up so well. For human PET data where nonbrain structures are not prominent, editing is probably not required.

If you do edit the data, you can choose thresholds of 1 (unless you want the threshold to provide some additional editing of low voxel values). Note that 8

and 16 bit data will require different thresholds and that the thresholds should be chosen to exclude nonbrain voxels.

The most common problem with the use of this algorithm is inappropriate selection of the thresholds. If you are using an eight bit version of AIR, a PET data threshold around 55 works well. For MRI data, a threshold around 10 is often but not always appropriate. For a sixteen bit version of AIR, a PET threshold around 14000 may be about right if the image uses the full dynamic range, but a proportionately lower threshold will be needed if only part of the range is utilized. MRI data often only uses 12 of the available 16 bits, so appropriate values typically will be in the 160-2560 range for 16 bit versions of AIR. It is best to look at the images to pick a threshold that excludes nonbrain regions.

The “Assume non-interaction of spatial parameter derivatives” option under Convergence tab in the Algorithm Parameters makes a substantial difference in terms of reducing registration time.

Linear

When choosing a spatial model, do not assume that more is better. While you can use a 15 parameter model to perform intrasubject registration, the results will be slower. Furthermore, unless there truly is some element of nonrigid-body distortion of the images, the extra parameters that you derive will be errors. If you know that your scanner systematically introduces some sort of linear distortion, the best approach would be to understand the distortion and systematically remove it before registration. However, if this is not practical, use of a model with more freedom does represent a reasonable alternative.

F.3.1.2 WARP

If you are using the algorithm for the first time, its best to start with a low order polynomial and work your way up to get a feel for how long the registration requires. Second order polynomial models run reasonably fast, but fifth order polynomials are extremely slow. You might also consider using very sparse sampling to get a feeling for speed (e.g., use 81 for both the Final and Initial Sampling).Termination and initialization files will allow you to proceed to higher order models without having to rederive any work already done. If you want to do a fifth order fit right away, set it up to run overnight (or maybe even over a weekend).

For PET data, you probably will only get a third order polynomial even if you request a fifth order because the algorithm generally can’t improve upon the third order results with PET data. Since fifth order fits take much longer, you might as well save a lot of time and only ask for third order in the first place. MRI data can generally sustain improvements through fifth order.

* Missing data due to a limited field of view can lead to unexpected or even bizarre results with high order warping. If you are dealing with a restricted field of view, you should probably stay with second or third order nonlinear models (or at least carefully inspect the results obtained with higher order models.

The first order polynomial registration provided by this algorithm is probably not as accurate as the one that can be derived using alignlinear. Although the spatial transformation models are identical, alignlinear takes advantage of the

invertibility of the transformation to compute the cost function in an unbiased fashion such that registration of image A to image B will be the exact inverse of registration of image B to image A. This is not the case here. In fact, I prefer to use alignlinear to derive the initial linear transformation using the scaled least squares cost function and then use that data to create an initialization file for this program starting with a second order transformation. However, note that the termination file created by alignlinear is not in the correct order for use as an initialization file by this program so you must convert it to the correct format.

The use of mask images by this program is different from alignlinear. There, the masks are applied in a way that assures that edited images are never directly compared to one another, preventing any tendency to just line up the edited edges. Here, the masks are applied to both images immediately, and the two edited images are directly compared in computing the cost function. Mask images are included here merely as a convenience. You will get the same results if you simply edit the images and register the edited versions.

Nonlinear transformations cannot be inverted analytically, so think carefully when deciding which image should be the standard image and which should be the reslice image.

F.4 RESLICE OPTIONS

Some interpolation models use scanline decomposition to accelerate interpolation. To minimize aliasing, scanline decomposition is preceded by oversampling of the data in a prepass interpolation step. AIR uses a conservative prepass interpolation that doubles the number of voxels along the necessary dimensions, a strategy that should be valid even with fairly large rotations (assuming cubic voxels and a rigid- body spatial transformation model). The issue of aliasing in scanline interpolation is complex and has not been fully addressed for 3D. For a discussion in 2D, see: Fraser, D, Schowengerdt, RA. *Avoidance of additional aliasing in multipass image rotations*. IEEE Transactions on Image Processing 1994;3:721-735.

The half window widths for sinc interpolation control the number of surrounding voxels that contribute to the interpolated value along each axis of the reslice file. A half-window width of 1 results in interpolation along that axis that only includes the nearest neighbor on either side and a half-window width of 6 will include the six nearest neighboring voxels in either direction. The total number of voxel included for standard 3D sinc interpolation is therefore

8*(x_half_window_width)*(y_half_window_width*(z_half_window_width)

.

The larger the half windows, the more closely the interpolation will match true sinc interpolation, but also the slower the resampling process.

Windowing of sinc interpolation is implemented using a Hanning window function exactly as described by Hajnal JV, Saeed N, Soar EJ, Oatridge A, Young IR, Bydder GM. Journal of Computer Assisted Tomography 1995:19:289-296.

Chirp-z interpolation provides high resampled image quality similar to sinc interpolation but is much faster because the resampling is done in the Fourier domain.

F.5 ERROR MESSAGES

- Failure in smoothing routine - The smoothing routine ran out of RAM. Try smoothing the file inside MEDx and then run AIR on the smoothed result without the smoothing option.
- WARNING: the voxel z_size differs for the two files that you are aligning using a 2D in-plane model - The registration may run without problem, but the reslicing may generate subsequent difficulties.
- WARNING: Hessian matrix is not positive definite... - The minimization algorithm has identified a potential problem with the cost function tending towards a maximum or being at a saddle point. Depending on whether you were close to or far from convergence at the time, the results may be acceptable or they may be bad. Either inspect the results closely, or consider using the "Assume non-interaction of spatial parameter derivatives" option which is less likely to give rise to a non-positive definite Hessian matrix.

Appendix F:
Automated Image Registration (Continued)

Note: The rest of the Appendix F has been take directly from the AIR 3.0 home page (http://bishopw.loni.ucla.edu/AIR3) Copyright 1995- 98 Roger P. Woods, M.D.(rwoods@ucla.edu) Some formatting of text has been changed.

F.6 General Information about AIR

The information provided in this section is relevant to anyone who plans to use the AIR package.

- Synopsis the programs
- Quantitation and registration
- Method of interpolation
- The problem of missing data
- Preferred image orientation
- Constraints for image dimensions and voxel sizes
- Preferred number of bits/pixel
- Preferred image resolution
- Overwriting of files

F.7 Synopsis of the Programs

The AIR 3.0 package includes two programs for automated registration of images. The first of these, alignlinear, includes 2D and 3D variants of all linear spatial transformation models, including rigid-body models, global rescaling models, and affine models. It includes all of the functionality of the old AIR 1.0 programs, alignmritopet, alignpettomri, and alignpettopet and can also be used for linear intersubject registration. It generates a file, referred to here as a ".air file", that contains linear transformation parameters for resampling one of the images to match the other. The second alignment program, align_warp, includes 2D and 3D variants of second, third, fourth, and fifth order nonlinear polynomial spatial transformation models. It generates a file, referred to here as a ".warp file", that contains nonlinear transformation parameters for resampling one of the images to match the other. Generally speaking, programs in the AIR package deal either with .air files or with .warp files, but not with both. Programs that deal with .warp files always include "_warp" at the end of their names.

The program reslice will use the registration parameters in a .air file to resample the reslice file (which is explicitly identified in the .air file) to match the standard file. Nearest neighbor, trilinear, or various forms of sinc interpolation can be used. The program reslice_warp provides similar functionality for .warp files.

The alignment programs depend upon the image headers for certain information about file dimensions and sizes. If you do not already have headers for your images, you can create them using the program makeaheader. The program scanheader will display the relevant information contained within an image header so that you can verify that the information is correct. The program fixheader will allow you to modify the real world sizes that correspond to the voxel sizes of an image if the header information about these sizes is inaccurate.

For use with AIR, individual slices from an image must be concatenated into a single image volume. The programs reunite and separate facilitate conversion between data formats that store each slice in a separate file and the multiimage format required by AIR.

The programs scanair and scan_warp allow you to display the information contained in .air and .warp files. Several programs are available to make specific modifications to .air and .warp files. The programs mv_air and mv_warp allow you to change the name of the reslice file designated in a preexisting .air or .warp file. This feature allows you to use the same parameters to resample multiple files that you know (or assume) to be spatially equivalent to one another. The programs cd_air and cd_warp allow you to update .air and .warp files to reflect the fact that the reslice file has been moved to some other directory in the file hierarchy. The same effect could be achieved using mv_air and mv_warp, but cd_air and cd_warp do not require you to keep track of the specific name of the reslice file (which is unmodified) and are therefore well suited to safely changing only the directory name. The program invert_air will exchange the reslice and standard files (adjusting the transformation matrix accordingly) so that you can use parameters originally derived for aligning file A to B to align file B to A instead. There is no counterpart program for .warp files due to the fact that nonlinear transformations cannot generally be analytically inverted. The program combine_air allows you to combine multiple .air files into a single .air file that will have the same effect as applying the individual files sequentially to a single data set, but without the accumulation of interpolation errors and loss of data outside the field of view that occurs with sequential resampling.The program combine_warp allows a single .warp file to be combined with .air files and is an exception to the general rule that programs do not deal with both .air and .warp files.

Because the .air and .warp files are stored separately from the image files, an additional level of file identification has been incorporated into these files. In addition to the names of the files, a ten digit identifier is computed based on the data in the file. This identifier is displayed along with the other information shown by scanair. In the event that you are unsure that a file is the file used to derive a set of registration parameters, you can use the program identify to compute the ten digit number that corresponds to a given data file.

Several programs are provided to manipulate the size or orientation of files without changing any voxel values. The program reorient allows you to rotate your data 90 degrees or 180 degrees around the x-, y-, or z- axis and also makes it possible to flip the data (effectively converting it to its mirror image) along any of these axes. The program resize will shift, clip or pad your data to achieve any desired matrix size. The program layout will composite multiple slices from an image into a single 2D file to create illustrations, etc.

Other programs reformat data in specified ways that do alter voxel values. The program zoomer will interpolate a study with anisotropic voxel sizes to generate a volume composed of cubic voxels. The program magnify uses Fourier interpolation to increase the number of voxels along a given image dimension. The program gsmooth will apply a Gaussian smoothing filter to a file.

The program manualreslice is the only interactive program in the AIR package. This program requires you to specify values for roll, pitch, yaw, x_shift, y_shift, and z_shift, x-axis_scaling, y-axis_scaling and z-axis_scaling. These values can then be stored in an initialization file for use with the linear automated alignment programs. Alternatively, these values can be used to manually generate a .air file or to manually reslice a file according to these parameters without generating a .air file. When used together with the capabilities of combine_air, and reslice, or with combine_warp and reslice_warp, manually created .air files make it possible for you to resample your data consistently with a pixel size and interplane distance of your choosing.

The program definecommon_air will average together a series of .air files to define an "average" common space for data analysis. The program reconcile_air will compare various .air files with redundant, potentially conflicting information and will create new .air files that are more internally consistent with one another.

The program softmean will take missing data into account in generating a mean image from resliced data that can be used for additional subsequent automated image registration.

Three programs are provided to manage binary files. These are useful for saving editing information, regions of interest, etc.To create a binary file from a non-binary file, use the program binarize. Binary files can be combined, intersected, subtracted, etc., from one another using the program binarymath. Finally, the program binarymask will apply a binary file to a non-binary file to create a masked non-binary file.

The program setheadermax will change the global maximum value in the header of 16 bit images so that they can be converted to 8 bit images with as many significant figures of precision as possible.

The program sizeof shows the size of various variable types generated by your C compiler and compares them to the corresponding sizes in the AIR development environment.

F.8 QUANTITATION AND REGISTRATION

With the AIR package, you should be able to reorient tomographic images of the same subject to match one another. If both images were acquired using the same modality, it is likely that you will want to perform a quantitative comparison of the two images to see if there has been any change between the two images. In making such comparisons, you should be aware of the fact that there are a number of issues that may arise when comparing a pair of reoriented images that do not arise when comparing a pair of images that were acquired in identical positions:

- Any errors in normalization of values from different imaging planes will lead to artifacts.
- Any spatial distortions in your imaging system will lead to artifacts.
- Any errors in measurement of voxel sizes will lead to artifacts.
- In PET scanning, misalignment of one of the emission studies relative to the transmission study will lead to artifacts.
- Unless you have truly isotropic image resolution in all three dimensions, the process of reorienting the images will misalign the axes of best and worst resolution leading to artifacts.
- Differences in partial volume effects related to discrete sampling of the signal will lead to artifacts.
- Post-reconstruction resampling of the data in the process of reslicing will lead to interpolation artifacts.
- Reorienting only one of the images can produce systematic biases in your data. If you always reslice condition B to match condition A and then perform some statistical test on the difference, a statistically significant finding could merely reflect the fact that the image from condition B was always reoriented rather than the fact that condition B was truly different from condition A.

Problems 1-3 are particularly amenable to quantitative analysis using phantoms, and it is strongly recommended that you perform such studies and fix the problems if you suspect that they are leading to errors in quantitation. Problems 4-7 are more difficult to deal with, and in some cases may be unavoidable. A particularly difficult situation can arise if some state or task of interest is significantly associated with a systematic variation in head position. In this case, it may not always be possible to separate the errors in quantitation due to problems 4-7 from true changes. For band-limited data, the use of sinc interpolation in AIR 2.0 may help to minimize resampling interpolation artifacts, but most tomographic data is not fully band-limited in three dimensions at present. Problem 8 can generally be avoided by randomizing or pseudorandomizing which image is to be resliced.

You should note that quantitative errors above are equally applicable to any method of analyzing misregistered data (e.g., the use of regions of interest) but that they are particularly evident when the images are analyzed on a pixel-by-pixel basis (which essentially amounts to extremely tiny ROI's).

With the exception of sinc interpolation for band-limited data, the AIR package does not incorporate any features intrinsically designed to deal with the quantitation problems described above. In some situations, on-line registration, which is not discussed or supported in this release but is discussed in the first paper describing the PET-PET registration method, minimizes many of these quantitative errors.

Note that the AIR package completely separates the derivation of the registration parameters from the final resampling of the data. Consequently, you are able to apply the spatial information contained in the .air files at any point in the data analysis that you like. On-line registration is one extreme, with the information being used to make modification even before data is acquired. However, it should also be possible to apply this information post acquisition, but prereconstruction, etc.

F.9 METHOD OF INTERPOLATION

The AIR package now supports nearest neighbor, trilinear, and various forms of sinc and chirp-z interpolation.

F.10 THE PROBLEM OF MISSING DATA

F.10.1 Data missing due to misregistration

When a misregistered file is resliced to create a registered file, a portion of the resulting file will generally be missing because it was outside the field of view in the original image. Particularly if the field of view is limited, the missing data may include voxels that are within the brain on the reference image. The reslicing programs in the AIR package will assign a voxel value of zero to the regions of missing data which will have linear edges for when using linear models but may have curved edges when using nonlinear spatial transformation models. Users unfamiliar with this possibility may be puzzled by the absence of data, especially since even a slight movement may suffice to lose an entire plane of data in the resliced images (AIR will not extrapolate outside the boundaries of the original image, no matter how trivial the excursion outside these boundaries).

If you have software originally developed for analyzing images that did not require realignment, you should review the consequences of assigning values of zero to missing data.

F.10.2 Registration of images with missing data

When performing registration, the AIR package cannot distinguish between voxels within the images that are zero because of missing data and voxels that are truly zero. Consequently, except in special situations, images to be registered with the AIR package should not have missing data within the brain (structures outside the brain such as scalp, skull, and dura may be missing due to recommended editing for MRI-PET registration or for intersubject registration). Consequently, it is generally best to avoid registering images that have already been resliced. Where possible, registration to the original images (which are assured not to include missing data) is preferred. In instances where one of the

registration targets needs to be an average of several images that have been coregistered, use of the program softmean to create the average image can help to assure that no voxels have missing data.

One situation where it may be important to edit the data to remove data within the brain is when brain pathology has led to extreme focal changes between acquisitions that are disrupting the registration process. In this situation, it is possible to force the alignlinear algorithm to perform a unidirectional fit by using the -p1 0 or -p2 0 flags. These flags allow one of the images to be edited to exclude areas that should not be included when deriving the registration parameters.

F.11 PREFERRED IMAGE ORIENTATION

There is no preferred or required image orientation for the AIR 3.0 package, so long as you are consistent (or at least able to keep track of your inconsistencies and consistently take them into account when using the programs). The package should not be expected to rotate an image by more than 45 degrees from its initialization parameters, and an increasing failure rate should be anticipated as the necessary alignment approaches this value. The package is incapable of recognizing that two images are mirror image versions of one another, a problem that is particularly likely to go unrecognized when the two images being registered start out in different formats. Don't forget that two data sets can be three dimensional mirror images of one another by virtue of having their planes ordered in opposite directions (top to bottom versus bottom to top). A utility program called reorient is provided to allow you to perform repositioning and mirror imaging as needed to get the images oriented for automated registration. Additional fine tuning of the initial registration can be provided when necessary by using an initialization file with the program alignlinear or align_warp.

F.12 CONSTRAINTS FOR IMAGE DIMENSIONS AND VOXEL SIZES

AIR 3.0 does not require voxel sizes to be isotropic along any dimension (AIR 1.0 required isotropic x- and y- pixel sizes). You should be able to reorient a coronally or sagitally acquired MRI scan and align it to a PET study without having to worry about the fact that the resulting pixel sizes are anisotropic.

The AIR package is designed to be extremely flexible with regard to the dimensions of your image. For practical purposes, you are limited only by the amount of contiguous RAM that is available for loading an image into memory. It is recommended that you not interpolate your original data to generate cubic (or approximately cubic) voxels prior to using the AIR package. The package has internal interpolation capabilities that are quite fast, avoiding the need to use extra disk space to save what is actually redundant information. You will also avoid the quantitation errors introduced by repeatedly reinterpolating the same data. The one exception to this recommendation is that if you are averaging together multiple realigned images with the intent of using the averaged value for subsequent registration, there may be some advantage to reslicing the realigned images to cubic voxels and averaging them as cubic voxels.

F.13 PREFERRED NUMBER OF BITS/PIXEL

The AIR package can be compiled to use either 8 or 16 bits/pixel for internal representation of voxels. Regardless of its internal representation, the AIR package will load images of either 8 bits/pixel or 16 bits/pixel. A bit shifting process is used to convert images into the specified internal representation. The AIR package will only generate output images with the number of bits/pixel that it uses for internal representation. For example, if the AIR package is compiled for 8 bits/pixel, it can register two images that have 16 bits/pixel and generate an appropriate .air file. If this .air file is used by the program reslice compiled for 8 bits/pixel, the original 16 bit file will be loaded and resliced, but the output image will be saved as 8 bits/pixel. Note however, that the same .air file that was generated by an 8 bits/pixel registration program can be utilized by a 16-bit version of reslice to generate a 16 bits/pixel resliced image. You should be aware of the fact that this process will remap the absolute pixel values (in a systematic way). If you are doing absolute quantitation, you will need to take this remapping of pixel values into account. Issues related to 16 bit images are discussed in detail in the technical notes.

The interconversion between different data formats is handled exclusively by the data loading subroutines, and the other components of the package generally do not have any way of determining what the format of the original data actually was. For 16 bit data, an additional complication is imposed by the fact that there are three different ways that 16 bit data may be represented externally. The AIR package uses the header global maximum and global minimum values to determine which of the three data types is being used. This issue is discussed in detail in the technical notes.

F.14 PREFERRED IMAGE RESOLUTION

This is an extremely complicated issue, and most of what is said here consists of empirical guidelines that should be adjusted by your own practical experience.

F.14.1 PET data (intrasubject intramodality)

Extremely noisy high resolution PET images generally take longer to register than smoother, lower resolution images. Furthermore, slight (subpixel size) misregistration of data that is actually already perfectly aligned can result in a small amount of additional smoothing due to the trilinear interpolation process used in the AIR package. This additional smoothing may make the slightly misregistered images "look" better aligned to the AIR package than the registered images. This effect is minimized when the images are already fairly smooth (see 1992 reference for further discussion of this problem). Finally, there are some theoretical reasons to suspect that registration may be particularly robust when the image smoothness is isotropic in all three dimensions.

With these generalizations in mind, we currently reconstruct our PET H215O images with the highest possible resolution (~6mm in-plane, 10 mm axially) and smooth them to isotropic resolution (10mm in all directions) either using gsmooth or using the internal smoothing capabilities of the program alignlinear before registering them to each other.

F.14.2 MRI data (intrasubject, intramodality)

Our validation work suggests that slight smoothing of MRI data with a 2 mm Gaussian filter improves the internal consistency of the resulting transformations. It is uncertain whether this also results in an improvement in true accuracy. If you are going to ultimately smooth the data more anyway, there is little reason to avoid smoothing in the registration process. If you plan to keep and analyze the data at high resolution, arguments can be made both for and against smoothing during registration (note that smoothing to derive registration parameters is independent of smoothing during the ultimate resampling of the data using those parameters).

F.14.3 Intermodality registration

For intermodality (e.g. MRI to PET) registration, there are theoretical arguments that the PET scans should have the highest possible resolution (assuming that the MRI resolution will be even higher still). Consequently, we realign our 6mm resolution PET images (using the registration parameters derived by smoothing them) to match one another and then average them all together using the program softmean to generate a high resolution, moderately noisy mean image which is then registered to the edited MRI scan. Using somewhat smoother images (e.g., FWHM 7-8 mm) still provides excellent results for MRI to PET registration (see 1993 reference). We do not smooth the MRI images.

F.14.4 Intersubject registration

Smoothed and unsmoothed data give quite similar results for intersubject registration.

F.15 OVERWRITING OF FILES

In general, the programs in the AIR package will not overwrite existing files unless you have explicitly granted permission to do so. However, there is an exception to this rule. The programs alignlinear and align_warp will always overwrite a preexisting file with the same name as the output file--practical experience has shown that the annoyance of having the program refuse to save the results of many minutes or even tens of minutes of iterative computation because a file with that name already existed outweighs the risk of losing data. A few safeguards have been built in: the programs will announce the fact that they intend to overwrite an existing file before starting the iterations, hopefully giving you sufficient time to type control-C to terminate the program if you want. Also the programs will reject any file name containing ".img" or ".hdr" as an output file name. To avoid any possibility of data loss, it is recommended that you always make a copy of data that could not be easily replaced if overwritten and that you store the data in a safe location in your directory where it cannot be accidentally overwritten by these programs.

F.16 HOW TO ... (PET-PET AND MRI-PET)

- How to verify voxel sizes and file dimensions

- How to register two PET studies
- How to interpolate your standard file to cubic voxels
- How to review the contents of a .air file
- How to invert a .air file to reverse the direction of reslicing
- How to align and average several PET studies in preparation for MRI-PET registration
- How to reorient an MRI image if it is upside down, backwards, and/or mirror imaged in comparison to your PET images
- How to align a PET study (or averaged PET study) to an MRI study
- How to reslice the original unedited MRI to match the PET study using registration parameters derived with an edited MRI
- How to reslice each of your original PET studies to match the MRI study

F.16.1 How to verify and voxel sizes and file dimensions

Step 1: 1. Use the program scanheader to review the header information that the AIR package will use to align and manipulate your studies. Example:

scanheader pet1

will display the header information for the study pet.img

Step 2: The AIR package provides a means for directly modifying voxel sizes (see program fixheader) If the values for the file dimensions or bits/pixel are incorrect, the program makeaheader can be used to make a new, corrected header.

F.16.2 How to register two PET studies

Step 1: Verify the voxel sizes and file dimensions of the studies

Step 2: Decide which study you want to reslice--this study will be called "pet2" in this example. The other study will be called "pet1"

Step 3: Use alignlinear to derive a registration parameter file (called pet2.airpet1 in this example).

Step 4: If your PET images are not very noisy:

alignlinear pet1 pet2 pet2.airpet1 -m 6

Step 5: If your PET images have voxels that are extremely anisotropic

alignlinear pet1 pet2 pet2.airpet1 -m 6 -z

Step 6: If your PET images are moderately noisy:

alignlinear pet1 pet2 pet2.airpet1 -m 6 -b1 5.0 5.0 0.0 -b2 5.0 5.0 0.0
If your PET images are extremely noisy with voxels that are extremely anisotropic:

alignlinear pet1 pet2 pet2.airpet1 -m 6 -b1 8.0 8.0 0.0 -b2 8.0 8.0 0.0 -z

This may take several minutes.

Step 7: Use reslice to reslice one file to match the other. The resliced file will be called rpet2 (or crpet2 if you want it interpolated to cubic voxels).

Step 8: If you want the reslice file to have the same voxel z_size and number of planes as the standard_pet file:

```
reslice pet2.airpet1 rpet2 -k
```

Step 9: If you want the reslice file interpolated to cubic voxels in addition to being resliced:

```
reslice pet2.airpet1 crpet2
```

F.16.3 How to interpolate your standard file to cubic voxels

Step 1: 1. Use zoomer to interpolate the standard pet to cubic voxels. The resulting file in this example will be called cpet1

```
zoomer pet1 cpet1
```

F.16.4 How to review the contents of a .air file

Step 1: 1. To review the contents of .air file pet2.airpet1, use scanair:

```
scanair pet2.airpet1
```

F.16.5 How to invert a .air file to reverse the direction of reslicing

Step 1: Review the contents of the .air file you want to invert

Step 2: Use invert_air to create a new .air file

```
invert_air pet2.airpet1 pet1.airpet2
```

Step 3: Use reslice to create the desired file

```
reslice pet1.airpet2 rpet1 -k
```

F.16.6 How to align and average several PET studies in preparation for MRI-PET registration

Step 1: Choose one of the studies to serve as the standard to which all the others will be registered (pet1) will be chosen here

Step 2: Derive registration parameters to reslice each study to this standard using alignlinear:

```
alignlinear pet1 pet1 pet1.airpet1 -m 6
alignlinear pet1 pet2 pet2.airpet1 -m 6
alignlinear pet1 pet3 pet3.airpet1 -m 6
alignlinear pet1 pet4 pet4.airpet1 -m 6
alignlinear pet1 pet5 pet5.airpet1 -m 6
alignlinear pet1 pet6 pet6.airpet1 -m 6
```

Step 3: (You may want to include the -b option with additional smoothing if the images are noisy or the -z option if the voxels are extremely anisotropic)

Step 4: Reslice each study to match the standard and interpolate them all to cubic voxels at the same time:

```
reslice pet1.airpet1 crpet1
reslice pet2.airpet1 crpet2
reslice pet3.airpet1 crpet3
reslice pet4.airpet1 crpet4
reslice pet5.airpet1 crpet5
reslice pet6.airpet1 crpet6
```

Step 5: Use softmean to create a mean image (called meanpet here) suitable for registration with an MRI

```
softmean meanpet n crpet1 crpet2 crpet3 crpet4 crpet5 crpet6
```

F.16.7 How to reorient an MRI image if it is upside down, backwards, and/or mirror imaged in comparison to your PET images

Step 1: Use your display package to decide what needs to be done.

Step 2: Use reorient to create a new MRI file in the position you want. In this example, the original upside-down and backwards MRI (ubmri) will be rotated 180° around the x-axis to create a properly oriented MRI (mri):

Step 3: reorient ubmri mri xx

F.16.8 How to align a PET study (or averaged PET study) to an MRI study

Step 1: Reorient the MRI image if needed.

Step 2: Use your image editing package to edit the MRI to remove the scalp, skull, and meninges to create an edited MRI file (emri)

Step 3: Decide whether you want to derive parameters to match the MRI to the PET or the PET to the MRI. (You can always use invert_air if you change your mind later, or if you want both).

Step 4: Use alignlinear to align the studies. In this example, the edited MRI (emri) will be aligned to an averaged PET (meanpet) to create a registration parameter file called emri.airmeanpet

```
alignlinear meanpet emri emri.airmeanpet -m 6 -p1 0 -p2 256 -t2 10
```

Step 5: Reslicing of the MRI to match the PET or the PET to match the MRI using the derived registration parameter file is completely analogous to reslicing of PET studies as described above.

F.16.9 How to reslice the original unedited MRI to match the PET study using registration parameters derived with an edited MRI

Step 1: If you used alignpettomri to derive the registration parameters, you will need to create an inverted registration parameter file using invert_air:

```
invert_air meanpet.airemri emri.airmeanpet
```

Step 2: Copy the registration parameter file to a new file (uemri.airmeanpet in this example).

```
cp emri.airmeanpet uemri.airmeanpet
```

Step 3: Use mv_air to change the file to be resliced to the uneditedmri (mri) in the registration parameter file:

```
mv_air uemri.airmeanpet mri
```

Step 4: Use reslice to create the new resliced MRI file (rmri) to match your PET file:

```
reslice uemri.airmeanpet rmri -k
```

F.16.10 How to reslice each of your original PET studies to match the MRI study

Step 1: Register each of your original PET studies to the study that will serve as the standard as described above.

Step 2: Average your PET studies to make a mean image as described above.

Step 3: Align the MRI study to the mean image as described above.

Step 4: If you used alignlinear to align the MRI to the PET study, apply invert_air to the resulting registration parameter file to get parameters to align the mean PET to the MRI.

Step 5: Use combine_air to combine the registration parameter file for aligning the mean PET to the MRI (meanpet.airemri) with the registration parameter files for aligning each of the PET studies to the standard PET to create new registration parameter files for directly reslicing the PET studies to match the MRI:

```
combine_air pet1.airemri n meanpet.airemri pet1.airpet1
combine_air pet2.airemri n meanpet.airemri pet2.airpet1
combine_air pet3.airemri n meanpet.airemri pet3.airpet1
combine_air pet4.airemri n meanpet.airemri pet4.airpet1
combine_air pet5.airemri n meanpet.airemri pet5.airpet1
combine_air pet6.airemri n meanpet.airemri pet6.airpet1
```

Step 6: In each case, the program will state that the parameters are valid only if meanpet is spatially equivalent to pet1 . It is if you have followed the examples as outlined here.

Step 7: Use reslice to generate the resliced files (rrpet1, rrpet2, etc.):

```
reslice pet1.airemri rrpet1 -k
reslice pet2.airemri rrpet2 -k
reslice pet3.airemri rrpet3 -k
reslice pet4.airemri rrpet4 -k
reslice pet5.airemri rrpet5 -k
reslice pet6.airemri rrpet6 -k
```

F.17 HOW TO ... (MRI-MRI)

- How to convert slice data into volume data
- How to verify voxel sizes and file dimensions

- How to register two MRI studies
- How to interpolate your standard file to cubic voxels
- How to review the contents of a .air file
- How to invert a .air file to reverse the direction of reslicing
- How to convert volume data back into slice data
- How to reorient an MRI volume if it is upside down, backwards, and/or mirror imaged

F.17.1 How to convert slice data into volume data

Step 1: Remove all header information contained within each slice data file and convert the slice data to either 8 bit or 16 bit integer values. The AIR package does not include file format converters for removing header information or otherwise converting proprietary formats. Some file format converts refer to the desired format as "raw" format. You can verify that the file does not contain any extra header information by computing the single slice data file's size in bytes as (x_dimension*y_dimension*bits/pixel)/8 and comparing this with the actual size of the file in bytes.

Step 2: Change the name of each slice data file so that it contains only a single period followed by img at the end of the file name (e.g., mri01.img).

Step 3: Use the program makeaheader to create a .hdr file corresponding to each .img slice data file. You need to know the x and y dimensions of each image (use a z-dimension of one for single slice data). You also need to specify the voxel sizes along each dimension (the z-dimension should be the interslice distance, which is not necessarily the same as the slice thickness). Finally, you must know the number of bits per pixel, and, for 16 bit data, how the data is stored. All of the files that are to be converted into a single volume must have identical dimensions and voxel sizes, so you can make one .hdr file and just copy it for the other images if you like.

Example:

makeaheader mri01 3 256 256 1 0.8 0.8 3.0

will create a file called mri1.hdr for a 256x256 plane of type 3 16 bit data with voxel dimensions of 0.8 mm (i.e., FOV=204.8) and an interslice distance of 3.0 mm.

Step 4: Use the program reunite to combine the individual slices into a single volume.

Example:

reunite mrivolume y mri01 mri02 mri03 mri04 mri05 mri06 mri07 mri08 mri09 mri10

will create files called mrivolume.hdr and mrivolume.img containing the data from the 10 specified slices, mri01.img, mri02.img ...

mri10.img. Note that the order in which the file names are entered dictates the ordering of the planes.

F.17.2 How to verify and voxel sizes and file dimensions

Step 1: Use the program scanheader to review the header information that the AIR package will use to align and manipulate your studies.

Step 2: Example:

scanheader mri01

will display the header information for the study mri01.img

Step 3: The AIR package provides a means for directly modifying voxel sizes (see program fixheader) If the values for the file dimensions or bits/pixel are incorrect, the program makeaheader can be used to make a new, corrected header.

F.17.3 How to register two MRI studies

Step 1: Verify the voxel sizes and file dimensions of the studies

Step 2: Decide which study you want to reslice--this study will be called "mri2" in this example. The other study will be called "mri1"

Step 3: Identify a voxel value in the studies that will reliably distinguish areas outside the body from values in the brain. In this example, a 16 bit value of 7000 will be used for both files.

Step 4: Use alignlinear to derive a registration parameter file (called mri2.airmri1 in this example).

Step 5: If you want to use the default smoothing:

ri1 -m 6 -x 3 -c 0.1 -t1 7000 -t2 7000

Step 6: If your MRI images have voxels that are extremely anisotropic

alignlinear mri1 mri2 mri2.airmri1 -m 6 -x 3 -c 0.1 -t1 7000 -t2 7000 -z

Step 7: If you want the registration to be based on slightly smoothed versions of the data:

alignlinear mri1 mri2 mri2.airmri1 -m 6 -x 3 -c 0.1 -t1 7000 -t2 7000 -b1 2.0 2.0 2.0 -b2 2.0 2.0 2.0

This may take several minutes.

Step 8: Use reslice to reslice one file to match the other. The resliced file will be called rmri2 (or crmri2 if you want it interpolated to cubic voxels).

Step 9: If you want the reslice file to have the same voxel z_size and number of planes as the standard_mri file:

reslice mri2.airmri1 rmri2 -k

Step 10: If you want the reslice file interpolated to cubic voxels in addition to being resliced:

reslice mri2.airmri1 crmri2

F.17.4 How to interpolate your standard file to cubic voxels

Step 1: 1. Use zoomer to interpolate the standard mri to cubic voxels. The resulting file in this example will be called cmri1

zoomer mri1 cmri1

F.17.5 How to review the contents of a .air file

Step 1: 1. To review the contents of .air file mri2.airmri1, use scanair:

scanair mri2.airmri1

F.17.6 How to invert a .air file to reverse the direction of reslicing

Step 1: Review the contents of the .air file you want to invert

Step 2: Use invert_air to create a new .air file

invert_air mri2.airmri1 mri1.airmri2

Step 3: Use reslice to create the desired file

reslice mri1.airmri2 rmri1 -k

F.17.7 How to convert volume data back into slice data

Step 1: 1. Use the program separate:

separate mrivolume smri y

This command will take the volume "mrivolume.img" and create a series of images "smri001.img", "smri002.img", "smri003.img" ... which are raw format data files containing one slice of data per file.

F.17.8 How to reorient an MRI volume if it is upside down, backwards, and/or mirror imaged

Step 2: Use your display package to decide what needs to be done.

Step 3: Use reorient to create a new MRI file in the position you want. In this example, the original upside-down and backwards MRI (ubmri) will be rotated 180° around the x-axis to create a properly oriented MRI (mri):

reorient ubmri mri xx

F.18 HOW TO ... (SUBJECT-SUBJECT)

- How to verify voxel sizes and file dimensions
- How to register subjects to one another or to an atlas using a linear affine spatial transformation
- How to create your own atlas
- How to register subjects to an atlas using nonlinear spatial transformations
- How to link together a series of .air files and a .warp file

F.18.1 How to verify and voxel sizes and file dimensions

Step 4: 1. Use the program scanheader to review the header information that the AIR package will use to align and manipulate your studies.

Example:

scanheader pet1

will display the header information for the study pet.img

Step 5: The AIR package provides a means for directly modifying voxel sizes (see program fixheader) If the values for the file dimensions or bits/pixel are incorrect, the program makeaheader can be used to make a new, corrected header.

F.18.2 How to register subjects to one another or to an atlas using a linear affine spatial tranformation

Step 1: If you are using MRI data, manually edit the data to remove nonbrain structures. For PET data with a PET atlas, editing is not required.

Step 2: In this example, the file called atlas can either be an edited image or averaged edited image in a standardized space (e.g., Talairach space) or it can simply be edited data from another subject in its native space.

Step 3: Use alignlinear to derive a registration parameter file (called mri01.airatlas in this example).

alignlinear atlas mri01 mri01.airatlas -m 12 -x 3 -c 1

This can take many minutes.

Step 4: Use reslice to reslice the mri01 to match the atlas. The resliced file will be called rmri01 (or cmri01 if you want it interpolated to cubic voxels).

Step 5: If you want the reslice file to have the same voxel z_size and number of planes as the standard_pet file:

reslice mri01.airatlas rmri01 -k

Step 6: If you want the reslice file interpolated to cubic voxels in addition to being resliced:

reslice mri01.airatlas crmri01

F.18.3 How to create your own atlas

Step 1: Pick one subject and register the images of all subjects to the one you have selected (register that subject to their self as well). This example will assume that you have registered three subjects to subject number one to generate .air files, "mri01.airmri01", "mri02.airmri01", and "mri03.airmri01".

Step 2: Use definecommon_air to create .air files for registering each individual into a space that approximates the average size, shape and orientation of the original images of the subjects:|

definecommon_air mri .airmri01 mri .aircommon y 01 02 03

This will produce output .air files, "mri01.aircommon", "mri02.aircommon" and "mri03.aircommon".

Step 3: Use reslice to resample each original file into the common space:

reslice mri01.aircommon crmri01
reslice mri02.aircommon crmri02
reslice mri03.aircommon crmri03

Step 4: Use softmean to average the files in the common space:

softmean atlas y null crmri01 crmri02 crmri03

This will create an averaged file called "atlas.img"

Step 5: If you like, you can repeat this entire process using "atlas" as the target in step 1 (don't register "atlas" to itself). You are unlikely to see much change after one or two iterations.

Step 6: If you want the atlas to have a standardized orientation (e.g., with the AC-PC line horizontal), you can figure out the transformation required to achieve this orientation and then use manualreslice to create a .air file that will allow "atlas" to be resliced to that orientation (use only rigid-body transformations, i.e.no rescaling). Suppose that this file is called "atlas.airstandardspace", you can then create .air files that will allow each individual file to be resliced directly into that standard space using combine_air:

combine_air mri01.airstandardspace y atlas.airstandard mri01.airatlas
combine_air mri02.airstandardspace y atlas.airstandard mri02.airatlas
combine_air mri03.airstandardspace y atlas.airstandard mri03.airatlas

Step 7: Now you can reslice each individual file directly into the standard space:

reslice mri01.airstandard crmri01 -o
reslice mri02.airstandard crmri02 -o
reslice mri03.airstandard crmri03 -o

Step 8: And create an average atlas in the standard space:

softmean standardatlas y null crmri01 crmri02 crmri03

The file "standardatlas.img" is the final atlas.

F.18.4 How to register subjects to an atlas using nonlinear spatial transformations

Step 1: If you are using MRI data, manually edit the data to remove nonbrain structures. For PET data with a PET atlas, editing is not required.

Step 2: In this example, the file called atlas can either be an edited image or averaged edited image in a standardized space (e.g., Talairach

space) or it can simply be edited data from another subject in its native space.

Step 3: Use align_warp to derive a registration parameter file (called mri01.warpatlas in this example) using a 5th order polynomial fit for MRI data (or a 3rd order polynomial fit for PET data).

Step 4: For MRI data:

```
align_warp mriatlas mri01 mri01.warpmriatlas -m 5
```

Step 5: For PET data:

```
align_warp petatlas pet01 pet01.warppetatlas -m 3
```

This can take a long time. You may get better results by initializing align_warp with a linear tranformation derived using alignlinear. See the align_warp page for details.

Step 6: Use reslice_warp to reslice the original file to match the atlas:

```
reslice_warp mri01.warpmriatlas rmri01 -k
```

F.18.5 How to link together a series of .air files and a .warp file

Suppose that you have:

- pet1.airmeanpet (.air file for registering scan "pet1" to a mean pet space)
- meanpet.airmri (.air file for registering the mean pet space to the subject's MRI)
- mri.warpatlas (.warp file for registering the subject's MRI to an atlas)
- atlas.airstandardspace (.air file for registering the atlas into some standardized space)

and you want a single file for transforming "pet1" directly into the standardized space.

Step 1: Use combine_air to combine any sequential .air files into a single tranformation:

```
combine_air pet1.airmri y meanpet.airmri pet1.airmeanpet
```

Step 2: Use combine_warp to combine .air and .warp files into a single .warp file:

```
combine_warp pet1.warpstandardspace y atlas.airstandardspace
mri.warpatlas pet1.airmri
```

Step 3: Use reslice_warp to resample "pet1.img"

```
reslice_warp pet1.warpstandardspace rpet1 -k
```

F.19 PROGRAMS IN THE AIR PACKAGE

The programs in the AIR package can be subdivided according to the types of files that they generate or modify.

- Programs are available that will:
  - generate new .air files
  - modify or combine .air files
  - generate new .warp files
  - modify or combine .warp files
  - generate or modify header files
  - generate new image files
  - display information about a file
  - create initialization files
  - use or create binary files
- Programs that will generate new .air files
  - alignlinear (all forms of registration)
  - manualreslice (trial and error registration)
  - resize (clipping, shifting or padding with no rotation)
  - alignmritopet (old AIR 1.0 algorithm, no longer distributed)
  - alignpettomri (old AIR 1.0 algorithm, no longer distributed)
  - alignpettopet (old AIR 1.0 algorithm, no longer distributed)
- Programs that will modify or combine .air files
  - cd_air (change target directory)
  - combine_air (combine multiple .air files)
  - combine_warp (combine .air and .warp files)
  - definecommon_air (average together .air files)
  - invert_air (interconvert reslice file and standard file)
  - mv_air (change reslice file)
  - reconcile_air (resolve inconsistencies between .air files)
- Programs that will generate new .warp files
  - align_warp (nonlinear registration)
- Programs that will modify or combine .warp files
  - cd_warp (change target directory)
  - combine_warp (combine .air and .warp files)

- mv_warp (change reslice file)

- Programs that will generate or modify header files
  - fixheader (correct header voxel sizes)
  - makeaheader (create new header)
  - setheadermax (change header maximum)
- Programs that will generate image files
  - layout (composite multiple slices onto a single 2D image)
  - magnify (Fourier interpolation to increase number of pixels)
  - manualreslice (arbitrary repositioning)
  - reorient (rotate 90 or 180 degrees or mirror image)
  - resize (clip, shift or pad data to change matrix size)
  - reslice (reposition based on .air file)
  - reslice_warp (reposition based on .warp file)
  - reunite (combine single slices into multivolume data set)
  - separate (split multivolume data into single slices)
  - softmean (average files together)
  - zoomer (interpolate file to cubic voxels)
- Programs that will display information about a file
  - identify (compute hash value, verify header and image compatibility)
  - scanair (display contents of .air file)
  - scanheader (display contents of header file)
  - scan_warp (display contents of .warp file)
- Programs that will create initialization files
  - manualreslice
- Programs that will use or create binary files
  - binarize (use threshold to convert non-binary file to binary file)
  - binarymask (apply binary file to mask non-binary file)
  - binarymath (combine or compare binary files to create new binary file)

F.20 PROGRAM INDEX

- alignlinear (AIR 3.0 automated linear registration both within and across subjects)
- alignmritopet (old AIR 1.0 intermodality registration, no longer distributed)
- alignpettomri (old AIR 1.0 intermodality registration, no longer distributed)
- alignpettopet (old AIR 1.0 intramodality registration, no longer distributed)
- align_warp (AIR 3.0 automated nonlinear registration)
- binarize (converts 8 or 16 bit file to a binary file)
- binarymask (applies a binary mask to a file)
- binarymath (performs binary operations on pairs of binary input files)
- cd_air (changes .air file's target directory)
- cd_warp (changes .warp file's target directory)
- combine_air (combines multiple .air files)
- combine_warp (combines .air and .warp files)
- definecommon_air (finds a good common space for data analysis based on .air files)
- fixheader (corrects header file's voxel sizes)
- gsmooth (Gaussian convolution)
- identify (computes hash value, verifies header and image compatibility)
- invert_air (interconverts .air file's reslice file and standard file)
- layout (creates 2D layout of 3D data)
- magnify (magnifies images using Fourier interpolation)
- makeaheader (creates new header files)
- manualreslice (trial and error registration, creates initialization files)
- mv_air (changes .air file's reslice file)
- mv_warp (changes .warp file's reslice file)
- reconcile_air (reconciles discrepancies between a series of .air files defining all possible pairwise registrations)
- reorient (rotates file 90 or 180 degrees or creates file's mirror image)
- resize (changes file's matrix size and/or shifts file within matrix)
- reslice (reslices data based on specified .air file)
- reslice_warp (reslices data based on specified .warp file)
- reunite (joins multiple 2D files into a single 3D file)
- scanair (displays contents of .air file)

- scanheader (displays contents of header file)
- scan_warp (displays contenst of .warp file)
- separate (cuts a 3D volume into multiple 2D files)
- setheadermax (changes header file's maximum)
- sizeof (displays C variable sizes)
- softmean (averages files together compensating for missing data)
- zoomer (interpolates file to cubic voxels)

F.21 PROGRAM INFORMATION

F.21.1 alignlinear

- purpose
- usage
- examples
- comments
- error messages
- see also

F.21.1.1 PURPOSE:

This is a general linear intramodality registration tool (within or across subjects, 2D or 3D). The user can specify any of a variety of models, including rigid-body, affine, or perspective.

The program will generate a .air file that can be used to reslice the specified reslice data set to match the specified standard data set.

F.21.1.2 USAGE:

alignlinear standard-file reslice-file air-out -m model_menu_number [options]

Model Menu:

- 3-D models:
  - 6. rigid body 6 parameter model
  - 7. global rescale 7 parameter model
  - 9. traditional 9 parameter model (std must be on AC-PC line)
  - 12. affine 12 parameter model
  - 15. perspective 15 parameter model
- 2-D models (constrained to in-plane, no interpolation):
  - 23. 2-D rigid body 3 parameter model
  - 24. 2-D global rescale 4 parameter model
  - 25. 2-D affine/fixed determinant 5 parameter model
  - 26. 2-D affine 6 parameter model

options:

- [-t1 threshold-standard-file]
- [-t2 threshold-reslice-file]
- [-b1 FWHM-x FWHM-y FWHM-z] (standard file)
- [-b2 FWHM-x FWHM-y FWHM-z] (reslice file)
- [-p1 partitions_in_standard_file]

- [-p2 partitions_in_reslice_file]
- [-e1 standard_file_mask]
- [-e2 reslice_file_mask]
- [-f initialization-file]
- [-g termination-file [overwrite?(y/n)]]
- [-w warp-format-termination-file [overwrite?(y/n)]]
- [-fs scaling_initialization file]
- [-gs scaling_termination_file [overwrite?(y/n)]]
- [-ws warp-format-scaling-termination-file [overwrite?y/n)]]
- [-s initial-sampling final-sampling sampling-decrement-ratio]
- [-c convergence-threshold]
- [-r repeated-iterations]
- [-h halt-after-(N)-iterations-without-improvement]
- [-a alternate-strategy-after-(M)-iterations-without-improvement]
- [-q] assume noninteraction of spatial parameter derivatives
- [-z] enable pre-alignment interpolation
- [-v] enable verbose mode
- [-x cost-function]

Cost functions:

- 1. standard deviation of ratio image
- 2. least squares
- 3. least squares with intensity scaling

where the following definitions apply:

`standard-file`

the name of the file that you want the other file resliced to match (.img or .hdr suffix optional)

`reslice-file`

the name of the file you want to reslice (.img or .hdr suffix optional)

`air-out`

the exact name of the output file (cannot contain '.img' or '.hdr' and will not be modified by the program).

`threshold-standard-file`

defines a minimum voxel value for the standard file. Voxels in the standard file below this value are excluded from analysis when computing the cost function and its derivatives in the forward direction. The value should always be an integer less than the maximum possible voxel value.

threshold-reslice-file

defines a minimum voxel value for the reslice file. Voxels in the reslice file below this value are excluded from analysis when computing the cost function and its derivatives in the reverse direction. The value should always be an integer less than the maximum possible voxel value.

FWHM-x FWHM-y FWHM-z (standard-file)

if this option is used, smoothing filters are applied along the x, y and z axes of the standard file before performing registration. The FWHM value specifies the full width at half maximum of the Gaussian smoothing filter to be applied along each dimension. The filters have units of millimeters (or whatever units you use to specify voxel sizes in your .hdr files). All three dimensions must be specified. If you give a value of zero, no smoothing will be applied along the corresponding dimension.

FWHM-x FWHM-y FWHM-z (reslice-file)

if this option is used, smoothing filters are applied along the x, y and z axes of the reslice file before performing registration. The FWHM value specifies the full width at half maximum of the Gaussian smoothing filter to be applied along each dimension. The filters have units of millimeters (or whatever units you use to specify voxel sizes in your .hdr files). All three dimensions must be specified. If you give a value of zero, no smoothing will be applied along the corresponding dimension.

partitions-in-standard-file

defines the number of partitions used for segmenting the standard file when computing the standard deviation of the ratio image in the forward direction. If this value is less than 1, no forward direction computation is performed. A value of 256 is typically used for intermodality registration if the standard file is an MRI study. A value of zero is typically used for intermodality registration if the standard file is a PET study. For intramodality registration, the default value of 1 is appropriate.

partitions-in-reslice-file

defines the number of partitions used for segmenting the standard file when computing the standard deviation of the ratio image in the reverse direction. If this value is less than 1, no reverse direction computation is performed. A value of 256 is typically used for intermodality registration if the reslice file is an MRI study. A value of zero is typically used for intermodality registration if the reslice file is a PET study. For intramodality registration, the default value of 1 is appropriate.

standard-file-mask

this file is appliedto the standard file as a mask. The file must match the standard file's dimensions, and voxels that are zero in this file will be excluded when computing the cost function in the forward direction. Mask files can be binary or regular files.

reslice_file_mask

this file is applied to the reslice file as a mask. The file must match the reslice file's dimensions, and voxels that are zero in this file will be excluded when computing the cost function in the reverse direction. Mask files can be binary or regular files.

initialization-file

the name of an ASCII file containing spatial transformation initialization parameters. These parameters can be used to control the starting position for automated registration, a feature that is useful if the initial misregistration is extreme (e.g., >45&degree; of rotational misregistration) or if the default registration leads to an obviously incorrect result. The format for the rigid-body initialization file is discussed under file types. Rigid-body initialization files are created most easily using the program manualreslice. Different spatial models require different numbers and types of parameters in the initialization file. Note that some cost functions may also allow an intensity parameter initialization file.

termination-file

the name of an ASCII file to be created containing spatial transformation termination parameters. These parameters can then be used as initialization parameters to restart the algorithm at the same location in parameter space where it left off (using the same spatial model). This allows you to switch cost functions or to vary smoothing, among other things. Different spatial models create incompatible files. Note that some cost functions may also allow an intensity parameter termination file.

overwrite?(y/n)

'y' if you want any preexisting file with the same name as your termination file to be overwritten.

warp-format-termination-file

the name of an ASCII file to be created containing spatial transformation termination parameters in a format compatible with the program align_warp. These parameters can be used as initialization parameters to start align_warp (with second order nonlinear terms) at the same location in parameter space where it left off. You may also want to create a warp format intensity parameter termination file.

scaling_initialization file

the name of an ASCII file containing the parameter that initializes intensity scaling. This is only applicable if the cost function includes intensity scaling as a formal parameter (i.e., the least squares with intensity scaling cost function).

scaling_termination_file

the name of an ASCII file to be created containing the intensity scaling parameter identified as optimal by the algorithm. This parameter can be used to restart the algorithm at the same location in parameter space where it left off (using the same spatial model). This is only applicable if the cost function includes intensity scaling as a formal parameter (i.e., the least squares with intensity scaling cost function).

warp_format_scaling_termination_file

the name of an ASCII file to be created containing the intensity scaling parameter identified as optimal by the algorithm. This parameter is in a format compatible with the program align_warp. Scaling termination files are only applicable if the cost function includes intensity scaling as a formal parameter (i.e., the least squares with intensity scaling cost function).

initial-sampling

controls how densely data is sampled during the first iterative cycle of the algorithm. Large values generally speed up the registration process because gross misregistration can be detected with fairly superficial sampling of the data. However, choosing an excessively large value can be counterproductive if the algorithm falls into an infinite loop or is led far from the true value by nonrepresentative sampling. Avoid multiples of two when choosing sampling parameters. If any of your matrix dimensions are divisible by two, the sampling will become spatially biased until the sampling density reaches one, at which point the algorithm will have to iteratively overcome the earlier bias at the maximal sampling density. If your matrix dimensions are divisible by three, you will have a similar problem with sampling densities that are multiples of three.

final-sampling

controls how densely data is sampled during the final iterative cycle of the algorithm. If your data is oversampled, the time spent sampling very densely may not provide any significant improvement in accuracy. In such cases, you may want to choose a final_sampling that is greater than one. Iterations will cease if the new sampling density is less than the final_sampling density specified here.

sampling-decrement-ratio

determines the number of intermediate iterative cycles of the algorithm. The current sampling is divided by this ratio with each cycle to determine the new sampling.

convergence-threshold

controls how small the predicted change in the cost function must be in order to meet the convergence criteria. Setting this value too large will result in convergence while the images are still misregistered; setting it too small may lead to a failure to converge.

repeated-iterations

controls the maximum number of iterations permitted at each sampling density. If this number is made too low, it will lead to inaccurate results and/or slow down the overall performance of the algorithm by preventing you from making use of information that could have been derived more quickly at the prematurely aborted, more superficial sampling.

halt-after-(N)-iterations-without-improvement

controls the maximum number of iterations without any observed improvement in the cost function. If greater than or equal to the "repeated_iterations" variable above, this value has no effect. At lower values, it can help you escape from situations where you are bouncing back and forth between two or three locations in parameter space without making any real progress. This sort of thing usually only happens at superficial sampling densities.

alternate-strategy-after-(M)-iterations-without-improvement

similar to the preceeding option except that it does not force termination of the current sampling density, but rather tries to split the difference between the locations in parameter space at the current sampling. If greater than or equal to the "halt-after-(N)-iterations-without-improvement" or the "repeated-interations" variables above, this value has no effect.

assume noninteraction of spatial parameter derivatives

ignoring the second derivatives of the interpolated voxel values with respect to spatial location by selecting this option can significantly improve the speed of the algorithm with little cost in terms of accuracy.

pre-alignment interpolation

in contrast to AIR 1.0, the current version of AIR does not apply prealignment interpolation of the files to cubic voxels by default. If you want prealignment interpolation, it can be enabled using this flag. Using prealignment interpolation will slow down the algorithm if you have thick slices, but can give more accurate results, especially with voxels that are extremely anisotropic. If your voxels are already cubic, prealignment interpolation has no effect.

verbose mode

information about every iteration will be printed to the screen if you use the -v option. If you run a lot of registrations and leave screen scrolling enabled with this option, you will eventually fill up the disk and the system will grind to a halt.

cost-function

determines which cost function is used for aligning the images. This should be a number from the corresponding menu:

1. standard deviation of ratio images

This cost function has the advantage of being independent of image intensity, so image intensities can be poorly matched and the registration will not be adversely affected. This is the only model that allows multiple partitions as required for intermodality registration.

2. least squares

This cost function assumes that the image intensities are scaled identically. Least squares is computationally simpler and therefore faster than the standard deviation of ratio images, but may be inaccurate if the image intensities are poorly matched.

3. least squares with intensity rescaling

This cost function is identical to the least squares cost function except that an intensity scaling term is added to the model.

menu-model

specifies the spatial model to be used to align the images. This should be a number from the corresponding menu:

6. rigid body 6 parameter model

Used for intra-subject registration when all voxel sizes are known accurately

7. global rescale 7 parameter model

Not too useful for biological data

9. traditional 9 parameter model (std must be on AC-PC line)

This is the typical (as opposed to literal) Talairach model, provided that the standard file has been properly oriented using the Talairach rules.

12. affine 12 parameter model

This is the preferred model for intersubject registration

15. perspective 15 parameter model

The perspective distortions are probably not worth the extra computational cost in most cases.

23. 2-D rigid body 3 parameter model

Analogous to the 6 parameter 3D model

24. 2-D global rescale 4 parameter model

Analogous to the 7 parameter 3D model. Might be useful for aligning photos of distant objects taken from various distances.

25. 2-D affine/fixed determinant 5 parameter model

This model allows for nonrigid distortion so long as total area is preserved. This may be a useful model for realigning data from serial tissue sections.

26. 2-D affine 6 parameter model

Analogous to the 12 parameter 3D model.

F.21.1.3 EXAMPLES:

alignlinear pet1 pet2 pet2.airpet1 -m 6 -t1 55 -t2 55 -x 1 -r 8 -c 0.0 -h 8 -a 8

- This will derive a .air file for aligning PET study pet2 to match PET study pet1. Thresholds of 55 will be set for each study, the standard deviation of ratio images cost function will be employed, a six parameter spatial model will be used and the algorithm will stop after eight iterations at each sampling density (using the default samplings of 81, 27, 9, 3, and 1). The total number of iterations will be the only applicable stopping criteria since the -h and -a flags have been effectively disabled by setting them equal to the -r flag and the -c flag has been disabled by setting it to zero.

F.21.1.4 COMMENTS:

- The most common problem with the use of this algorithm is inappropriate selection of the thresholds. If you are using an eight bit version of AIR, a PET data threshold around 55 works well. For MRI data, a threshold around 10 is often but not always appropriate. For a sixteen bit verson of AIR, a PET threshold around 14000 may be about right if the image uses the full dynamic range, but a proportionately lower threshold will be needed if only part of the range is utilized. MRI data often only uses 12 of the available 16 bits, so appropriate values typically will be in the 160-2560 range for 16 bit

versions of AIR. It is best to look at the images to pick a threshold that excludes nonbrain regions.

- When choosing a spatial model, do not assume that more is better. While you can use a 15 parameter model to perform intrasubject registration, the results will be slower. Furthermore, unless there truly is some element of nonrigid-body distortion of the images, the extra parameters that you derive will be errors. If you know that your scanner systematically introduces some sort of linear distortion, the best approach would be to understand the distortion and systematically remove it before registration. However, if this is not practical, use of a model with more freedom does represent a reasonable alternative.
- It is better to use mask files than to simply edit the data prior to registration. If you edit prior to registration, there will be a tendency to line up the edges created by editing which may allow the accuracy of the editing to become a predominant factor in determining the accuracy of the registration. When you specify a mask file, the program actually stores two versions of each file, one with editing and the other without. When the cost function is computed, it is always then based on an edited version of one image (dictating that edited regions do not contribute to the cost function) and an unedited version of the other (assuring that data is being compared to data, not to zeros created by editing).
- If you are having frequent problems with an error indicating the Hessian matrix is not positive definite, try using the -q option. The non-positive definite Hessian matrix is especially likely to arise when you try to register a file to a resliced version of itself (as people often do when they first try out the algorithm). In this particular situation, the problem is created by the fact that the two files only differ by interpolation and round-off errors which do not have well behaved second derivatives.

F.21.1.5 ERROR MESSAGES: (ALPHABETICAL BY CASE)

See also: Generic error messages

➢ A positive decimal number must follow -c
self explanatory

➢ A positive integer must follow -a
self explanatory

➢ A positive integer must follow -h
self explanatory

➢ A positive integer must follow -r
self explanatory

➢ A termination file name must follow -g
you must supply a valid file name after this argument

➢ A valid cost function number from the menu must follow -x
you must supply one of the cost function integers listed in the menu

➢ A valid model number from the menu must follow -m

you must supply one of the model integers listed in the menu

- An initialization parameter file name must follow -f

you must supply the name of a valid initialization file after this argument

- An integer must follow -t1 or -t2

self explanatory

- Argument after -a cannot start with - ; it must be a positive integer

self explanatory

- Argument after -c cannot start with - ; it must be a positive decimal number

self explanatory

- Argument after -f cannot start with - ; it must be an initialization parameter file name

you must supply the name of a valid initialization file after -f

- Argument after -g cannot start with -; it must be a termination file name

you must supply a valid name for a termination file

- Argument after -h cannot start with - ; it must be a positive integer

self explanatory

- Argument after -m cannot start with - ; it must be a valid model number from the menu

self explanatory

- Argument after -r cannot start with - ; it must be a positive integer

self explanatory

- Argument after -x cannot start with -; it must be a valid cost function number from the menu

self explanatory

- Attempt to save .air file ____ failed.

The .air file could not be saved. Do you have write permission on the disk? Is the disk full?

- Failure in smoothing routine

The smoothing routine ran out of RAM. Try smoothing the file externally using gsmooth and then run this program on the smoothed result without the -b option.

- File '____' exists, no permission to overwrite

You have not granted overwrite permission for the termination file that follows the -g flag and a file already exists with the name that you have specified. If you want to give overwrite permission, add 'y' after the file name in the command line.

- First three arguments cannot begin with a -

Self explanatory

- Name of output .air file cannot contain ____ or ____

This protects you from accidentally overwriting image data with a .air file. Choose a new name that does not contain the specified suffixes.

- Name of termination file cannot contain ____ or ____

This protects you from accidentally overwriting image data with a termination file. Choose a new name that does not contain the specified suffixes.

- Sorry, flag -____ is not defined for this program

You have specified a flag that is not defined. Review the defined flags and respecify.

- Sorry, model ____ is not defined

The spatial model you have selected does not exist. Review the menu of implemented models.

- Sorry, you have selected an unimplemented model for this cost function, please try again

The spatial model you have requested is not implemented for all cost functions. Choose an alternative model or cost function.

- Sorry, you have selected an unimplemented model, please try again

The spatial model you have selected does not exist. Review the menu of implemented models.

- Standard (____) and reslice (____) files both have only a single plane of data. You must use a 2D model in this situation

Select a 2D model (all 2D model numbers start with the digit '2').

- Standard file (____) has ____ planes and reslice file (____) has ____ planes

For 2D models, you must have an identical number of planes in the two image sets being registered. Data from any given plane of the reslice file is always mapped to data on the corresponding plane of the standard file.

- The initialization file only provided ____ parameters, 12 parameters are required for the affine model (up to 16 parameters can be specified, but parameters 13-16 must be 0 0 0 1

You can review and modify the initialization file with your favorite text editor.

- The initialization file only provided ____ parameters, 15 parameters are required for the perspective model (up to 16 parameters can be specified

You can review and modify the initialization file with your favorite text editor.

- The initialization file only provided ____ parameters, ____ are required for this model

You can review and modify the initialization file with your favorite text editor.

- The initialization file only provided ____ parameters, ____ are required for this model

You can review and modify the initialization file with your favorite text editor.

- The number of planes in the standard and reslice files must be identical for 2D alignment models

Self explanatory.

- The sixteenth parameter for the perspective model must be non-zero

You can review and modify the initialization file with your favorite text editor.

- Three arguments that follow -s cannot start with - ; they must be positive integers

Self-explanatory

- Three integers must follow -s

Self-explanatory

- Three positive numbers must follow -b1 or -b2

Self-explanatory

- WARNING: File '____' will be overwritten by the output of this program

Type control-C immediately to abort program if you don't want to overwrite this file

- WARNING: The initialization file contained more parameters (____) than required (____)

You can review this file using a text editor.

- WARNING: Hessian matrix is not positive definite...

The minimization algorithm has identified a potential problem with the cost function tending towards a maximum or being at a saddle point. Depending on whether you were close to or far from convergence at the time, the results may be acceptable or they may be bad. Either inspect the results closely, or consider using the -q option which is less likely to give rise to a non-positive definite Hessian matrix.

- WARNING: the voxel z_size differs for the two files that you are aligning using a 2D in-plane model

The registration may run without problem, but the resulting .air file may generate subsequent difficulties.

- WARNING: Your request for a rescaling termination file will be ignored because you have selected a model that does not use a rescaling parameter

Only the least squares cost function with intensity rescaling (-x 3) will generate a rescaling termination file.

- When the initialization file specifies >12 parameters for the affine model, parameters 13-16 must be 0 0 0 1

self-explanatory

- You have requested an unimplemented cost function

Choose a diffent cost function from the menu

- You must use a 2D model in this situation

Choose a 2D model from the menu.

- final_sampling (2nd argument after -s) cannot be > initial_sampling (1st argument after -s)

respecify arguments accordingly

- sampling_decrement_ratio (3rd argument after -s) must be > 1

respecify argument accordingly

- threshold ____ is not in range of possible pixel values

Choose a value less than 255 if you are using 8 bits/pixel on a SPARCstation

- unable to create termination file '____'

Do you have write permission and free space in the specified directory?

- unable to open initialization file '____'

Check to see that the file exists and that you have read permission

- unable to parse arguments,argument ____ was expected to begin with a -

review command line looking for an extra argument.

F.21.2 align_warp

- purpose
- usage
- examples
- comments
- error messages
- see also

F.21.2.1 PURPOSE:

This is a nonlinear registration tool that can be used within or across subjects and includes implementation of 2D and 3D nonlinear spatial transformation models.

The program will generate a .warp file that can be used to reslice the specified reslice data set to match the specified standard data set.

F.21.2.2 USAGE:

align_warp standard-file reslice-file .warp-out **-m** model-menu-number [final-model-menu-number] [options]

Model Menu:

- 3-D models:
  - 1. first order linear 12 parameter model
  - 2. second order nonlinear 30 parameter model
  - 3. third order nonlinear 60 parameter model
  - 4. fourth order nonlinear 105 parameter model
  - 5. fifth order nonlinear 168 parameter model
- 2-D models:
  - 21. first order linear 6 parameter model
  - 22. second order nonlinear 12 parameter model

- 23. third order nonlinear 20 parameter model
- 24. fourth order nonlinear 30 parameter model
- 25. fifth order nonlinear 42 parameter model

options:

- [-t1 threshold_standard_file]
- [-t2 threshold_reslice_file]
- [-b1 FWHM-x FWHM-y FWHM-z] (standard file)
- [-b2 FWHM-x FWHM-y FWHM-z] (reslice file)
- [-e1 standard_file_mask]
- [-e2 reslice_file_mask]
- [-f initialization-file]
- [-g termination-file [overwrite?(y/n)]]
- [-fs scaling_initialization file]
- [-gs scaling_termination_file [overwrite?(y/n)]]
- [-s initial-sampling final-sampling sampling-decrement-ratio]
- [-c convergence-threshold]
- [-r repeated-iterations]
- [-h halt-after-(N)-iterations-without-improvement]
- [-a alternate-strategy-after-(M)-iterations-without-improvement]
- [-d] don't write extra zeros (for higher order model) in termination file
- [-q] assume noninteraction of spatial parameter derivatives
- [-v] enable verbose mode

where the following definitions apply:

`standard-file`

the name of the file that you want the other file resliced to match (.img or .hdr suffix optional). The standard file will often be an atlas, but can also be images from a single subject.

`reslice-file`

the name of the file you want to reslice (.img or .hdr suffix optional)

`.warp-out`

the exact name of the .warp transformation parameter output file (cannot contain '.img' or '.hdr' and will not be modified by the program).

`model-menu-number`

The order (a number from 1 through 5) of the polynomial transformation used as a spatial transformation model. If the optional final-model-menu-number (see next item) is provided, the model selected here will be the model used initially,

and models will increment by one until the final model is reached. If the optional final-model-menu-number is not provided, the algorithm will start with a first order transformation and increment to the order specified by this argument.

final-model-menu-number

See the description of the model-menu-number above.

threshold-standard-file

defines a minimum voxel value for the standard file. Voxels in the standard file below this value are excluded from analysis when computing the cost function and its derivatives. The value should always be an integer less than the maximum possible voxel value.

threshold-reslice-file

defines a minimum voxel value for the reslice file. Voxels in the reslice file below this value are excluded from analysis when computing the cost function and its derivatives. The value should always be an integer less than the maximum possible voxel value.

FWHM-x FWHM-y FWHM-z (standard-file)

if this option is used, smoothing filters are applied along the x, y and z axes of the standard file before performing registration. The FWHM value specifies the full width at half maximum of the Gaussian smoothing filter to be applied along each dimension. The filters have units of millimeters (or whatever units you use to specify voxel sizes in your .hdr files). All three dimensions must be specified. If you give a value of zero, no smoothing will be applied along the corresponding dimension.

FWHM-x FWHM-y FWHM-z (reslice-file)

if this option is used, smoothing filters are applied along the x, y and z axes of the reslice file before performing registration. The FWHM value specifies the full width at half maximum of the Gaussian smoothing filter to be applied along each dimension. The filters have units of millimeters (or whatever units you use to specify voxel sizes in your .hdr files). All three dimensions must be specified. If you give a value of zero, no smoothing will be applied along the corresponding dimension.

standard-file-mask

this file is applied to the standard file as a mask. The file must match the standard file's dimensions, and voxels that are zero in this file will be treated as if they were zero in the standard file when computing the cost function. Mask files can be binary or regular files.

reslice_file_mask

this file is applied to the reslice file as a mask. The file must match the reslice file's dimensions, and voxels that are zero in this file will be treated as if they were zero in the reslice file when computing the cost function. Mask files can be binary or regular files.

initialization-file

the name of an ASCII file containing spatial transformation initialization parameters. If you override the default procedure of starting with a first order model, it is very important to use an initialization file (presumably derived from some prior successful registration). Poor initialization of high order transformations can lead to poor results. Each spatial model requires different numbers of parameters in the initialization file. Note that voxel intensity initialization is performed separately in an intensity parameter initialization file.

termination-file

the name of an ASCII file to be created containing spatial transformation termination parameters. These parameters can then be used as initialization parameters to restart the algorithm at the same location in parameter space. Unlike the program alignlinear, in align_warp the termination file automatically writes out an initialization file for the next higher order of spatial transformation by adding an appropriate number of zeros at the appropriate locations. If you don't actually want to increment the model, use the -x argument as well. Note that voxel intensity termination parameters must be requested separately by identifying an intensity parameter termination file.

overwrite?(y/n)

'y' if you want any preexisting file with the same name as your termination file to be overwritten.

scaling_initialization file

the name of an ASCII file containing the parameter that initializes intensity scaling.

scaling_termination_file

the name of an ASCII file to be created containing the intensity scaling parameter identified as optimal by the algorithm. This parameter can be used to restart the algorithm at the same location in parameter space where it left off. In addition, the scaling parameter could be used as an intensity normalization factor for subsequent statistical analysis of the registered data.

initial-sampling

controls how densely data is sampled during the first iterative cycle of the algorithm. Large values generally speed up the registration process because gross misregistration can be detected with fairly superficial sampling of the data. However, choosing an excessively large value can be counterproductive if the algorithm falls into an infinite loop or is led far from the true value by nonrepresentative sampling. Avoid multiples of two when choosing sampling parameters. If any of your matrix dimensions are divisible by two, the sampling will become spatially biased until the sampling density reaches one, at which point the algorithm will have to iteratively overcome the earlier bias at the maximal sampling density. If your matrix dimensions are divisible by three, you will have a similar problem with sampling densities that are multiples of three.

final-sampling

controls how densely data is sampled during the final iterative cycle of the algorithm. If your data is oversampled, the time spent sampling very densely may not provide any significant improvement in accuracy. Unlike alignlinear, the

default here is to end with sparse sampling at every ninth voxel. Iterations will cease if the new sampling density is less than the final_sampling density specified here.

sampling-decrement-ratio

determines the number of intermediate iterative cycles of the algorithm. The current sampling is divided by this ratio with each cycle to determine the new sampling.

convergence-threshold

controls how small the predicted change in the cost function must be in order to meet the convergence criteria. Setting this value too large will result in convergence while the images are still misregistered; setting it too small may lead to a failure to converge.

repeated-iterations

controls the maximum number of iterations permitted at each sampling density. If this number is made too low, it will lead to inaccurate results and/or slow down the overall performance of the algorithm by preventing you from making use of information that could have been derived more quickly at the prematurely aborted, more superficial sampling.

halt-after-(N)-iterations-without-improvement

controls the maximum number of iterations without any observed improvement in the cost function. If greater than or equal to the "repeated_iterations" variable above, this value has no effect. At lower values, it can help you escape from situations where you are bouncing back and forth between two or three locations in parameter space without making any real progress. This sort of thing usually only happens at superficial sampling densities.

alternate-strategy-after-(M)-iterations-without-improvement

similar to the preceding option except that it does not force termination of the current sampling density, but rather tries to split the difference between the locations in parameter space at the current sampling. If greater than or equal to the "halt-after-(N)-iterations-without-improvement" or the "repeated-interations" variables above, this value has no effect.

don't write extra zeros (for higher order model) in termination file

this will prevent the program from appending zeros to a termination file. This will allow the termination file to be reused with the same spatial transformation model rather than with the next higher order model.

assume noninteraction of spatial parameter derivatives

ignoring the second derivatives of the interpolated voxel values with respect to spatial location by selecting this option can significantly improve the speed of the algorithm with little cost in terms of accuracy.

verbose mode

information about every iteration will be printed to the screen if you use the -v option. If you run a lot of registrations and leave screen scrolling enabled with

this option, you will eventually fill up the disk and the system will grind to a halt.

F.21.2.3 EXAMPLES:

align_warp subject1 subject2 subject2.warpsubject1 -m 2 -t1 10 -t2 10 -g s2.init -gs s2.inits

- This will derive a .warp file for aligning an image of subject2 to match an image of subject1. Thresholds of 10 are applied to each file, and the algorithm will start with a first order transformation using the default assumption that the centers of the files should be aligned. The algorithm will automatically proceed to a second order polynomial spatial transformation model. Termination files will called s2.init and s2.inits will be created. If files called s2.init or s2.inits already exist, the program won't run because it doesn't have permission to overwrite these files.

align_warp subject1 subject2 subject2.warpsubject1 -m 3 3 -t1 10 -t2 10 -f s2.init -fs s2.inits -g s3.init y -gs s3.inits y

- This example picks up where the previous one left off. It reads the parameters of the previous model from the termination files and performs registration using a third order polynomial spatial transformation model. Termination files called s3.init and s3.inits are created. The .warp file from the first registration is overwritten.If files called s3.init or s3.inits already exist, they will be overwritten.

align_warp subject1 subject2 subject2.warpsubject1 -m 3 5 -t1 10 -t2 10 -f s3.init -fs s3.inits

- This example again picks up where the previous one left off. It reads the parameters of the previous model from the termination files and performs registration using a fourth and then a fifth order polynomial spatial transformation model. It does not create a termination file. The .warp file from the previous third order registration is overwritten.

F.21.2.4 COMMENTS:

- For MR data, it is recommended that you edit the data to remove nonbrain structures (e.g., scalp, skull and dura). Even if the algorithm does run successfully, you will have invested a lot of computation time in making sure that your subjects' noses are of similar size and shape, even if this means that their cerebellums don't line up so well. For human PET data where nonbrain structures are not prominent, editing is probably not required.
- If you do edit the data, you can choose thresholds of 1 (unless you want the threshold to provide some additional editing of low voxel values). Note that 8 and 16 bit data will require different thresholds and that the thresholds should be chosen to exclude nonbrain voxels.
- If you are using the algorithm for the first time, its best to start with a low order polynomial and work your way up to get a feel for how long the registration requires. Second order polynomial models run reasonably fast, but fifth order polynomials are extremely slow. You might also consider using very sparse sampling to get a feeling for speed (e.g.,add " -s 81 81 3" to your command line).Termination and initialization files will allow you to

proceed to higher order models without having to rederive any work already done. If you want to do a fifth order fit right away, set it up to run overnight (or maybe even over a weekend).

- For PET data, you probably will only get a third order polynomial even if you request a fifth order because the algorithm generally can't improve upon the third order results with PET data. Since fifth order fits take much much longer, you might as well save a lot of time and only ask for third order in the first place. MRI data can generally sustain improvements through fifth order.
- Missing data due to a limited field of view can lead to unexpected or even bizarre results with high order warping. If you are dealing with a restricted field of view, you should probably stay with second or third order nonlinear models (or at least carefully inspect the results obtained with higher order models.
- The -q option makes a substantial difference in terms of reducing registration time.
- The first order polynomial registration provided by this algorithm is probably not as accurate as the one that can be derived using alignlinear. Although the spatial transformation models are identical, alignlinear takes advantage of the invertibility of the transformation to compute the cost function in an unbiased fashion such that registration of image A to image B will be the exact inverse of registration of image B to image A. This is not the case here. In fact, I prefer to use alignlinear to derive the initial linear transformation using the scaled least squares cost function (-x 3 flag) and then use that data to create an initialization file for this program starting with a second order transformation. It is now possible to have alignlinear create a termination file that is properly formatted for use as an initialization file with this program (use the alignlinear -w option, possibly also with the -ws option). Alternatively, if you already used the -g option with alignlinear, you can convert it to the correct format.
- The use of mask files by this program is different from alignlinear. There, the masks are applied in a way that assures that edited images are never directly compared to one another, preventing any tendency to just line up the edited edges. Here, the masks are applied to both images immediately, and the two edited images are directly compared in computing the cost function. Maskfiles are included here merely as a convenience. You will get the same results if you simply edit the images and register the edited versions.
- Nonlinear transformations cannot be inverted analytically, so think carefully when deciding which file should be the standard file and which should be the reslice file.

F.21.2.5 ERROR MESSAGES: (ALPHABETICAL BY CASE)

See also: Generic error messages

➢ An image file name must follow -e1 or -e2

These are flags that indicate that the next argument is a mask file.

➢ A positive decimal number must follow -c

self-explanatory

- A positive integer must follow -a

self-explanatory

- A positive integer must follow -h

self-explanatory

- A positive integer must follow -r

self-explanatory

- A termination file name must follow -g

-g is a flag indicating that the name of an ASCII file to be created will follow

- A valid model number from the menu must follow -m

valid models are: 1, 2, 3, 4, 5, 21, 22, 23, 24, and 25

- An initialization parameter file name must follow -f

-f is a flag indicating that the name of an existing ASCII file will follow

- An integer must follow -t1 or -t2

self-explanatory

- Argument after -a cannot start with - ; it must be a positive integer

self-explanatory

- Argument after -c cannot start with - ; it must be a positive decimal number

-c is followed by the convergence threshold, it must be nonnegative

- Argument after -f cannot start with - ; it must be an initialization parameter file name

-f is a flag indicating that the name of an existing ASCII file will follow

Argument after -g cannot start with -; it must be a termination file name

-g is a flag indicating that the name of an ASCII file to be created will follow

- Argument after -h cannot start with - ; it must be a positive integer

self-explanatory

- Argument after -m cannot start with - ; it must be a valid model number from the menu

valid models are: 1, 2, 3, 4, 5, 21, 22, 23, 24, and 25

- Argument after -r cannot start with - ; it must be a positive integer

-r is a flag that should be followed by the maximum number of iterations at each sampling density

- Attempt to save .warp file ____ failed.

Do you have write permission for the specified directory?

Is the disk full?

- Dimension mismatch: ____: ____ ____ ____, ____: ____ ____ ____

Mask files must have the same dimension as the corresponding data files.

- Failure in smoothing routine

Inadequate RAM memory is the most likely cause

- File '____' exists, no permission to overwrite

Termination files will not overwrite existing files of the same name unless the termination file name is followed by a 'y'

- First three arguments cannot begin with a -

the first three arguments are reserved for the names of the standard file, reslice file and output file.

- Initial and final models must have the same dimension (2D or 3D)

You can't start with a 2D model and end with a 3D model or vice versa. Check the two arguments after -m

- Name of output .air file cannot contain .hdr or .img

self-explanatory--this is to prevent overwriting of your original data.

- Name of termination file cannot contain .hdr or .img

self-explanatory--this is to prevent overwriting of your original data.

- Sorry, flag -____ is not defined for this program

The specified flag does not have meaning to this program.

- Sorry, model ____ is not defined

The specified model is not implemented in this program.

- Standard (____) and reslice (____) files both have only a single plane of data ...

A 2D model is required in this case

- The final model cannot be lower than the initial model

The first argument after -m must be lower than the second argument after -m

- The initialization file provided ____ parameters, ____ are required for this model

The number of parameters in the initialization file must exactly match the number required by the selected model. Use your text editor to revise the initialization file.

- Three arguments that follow -s cannot start with - ; they must be positive integers

self-explanatory

- Three integers must follow -s

self-explanatory

- Three positive numbers must follow -b1 or -b2

These are the FWHM smoothing values for the x, y, and z dimensions of the files.

- WARNING: File '____' will be overwritten by the output of this program

The .warp file that you have specified will overwrite an existing file with the same name. Hit Control-C to terminate the program if overwriting is undesirable.

- WARNING: It is recommended that you use initialization files with

nonlinear transformations

You have requested that the algorithm start with a higher order (i.e.,nonlinear model) and have not provided an initialization file. You risk seeing the evil "Hessian matrix is not positive definite" warning and/or getting a bad result.

- WARNING: Hessian matrix is not positive definite...

The minimization algorithm has identified a potential problem with the cost function tending towards a maximum or being at a saddle point. Depending on whether you were close to or far from convergence at the time, the results may be acceptable or they may be bad. Either inspect the results closely, or consider using the -q option which is less likely to give rise to a non-positive definite Hessian matrix.

- You cannot use the -e1 flag twice on the same command line

You have requested two different masks be applied to the same data set.

- You cannot use the -e2 flag twice on the same command line

You have requested two different masks be applied to the same data set.

- You must specify a spatial transformation model using the -m argument

The -m argument, followed by a model number, is mandatory.

- Your final model ____ is not defined on the menu

self-explanatory

- final_sampling (2nd argument after -s) cannot be > initial_sampling (1st argument after -s)

self-explanatory

- sampling_decrement_ratio (3rd argument after -s) must be > 1

self-explanatory

- threshold ____ is not in range of possible pixel values

The threshold must be in the possible range for the data type of the corresponding file. If the threshold seems right, review the data in the header of the data file.

- unable to create termination file '____'

Do you have the necessary write permissions?

Is the disk full?

- unable to open initialization file '____'

Does the file exist?

Do you have read permission?

- unable to parse arguments,argument ____ was expected to begin with a -

You have a number or letter where a new argument flag was expected.

F.21.3 reslice

- purpose
- usage
- examples
- comments
- error messages
- see also
- references

F.21.3.1 PURPOSE:

This is the program that takes .air files and uses the information that they contain to load the corresponding image file and generate a new, realigned file.

F.21.3.2 USAGE:

reslice.air-file output [options]

options:

- [-a alternate-reslice-file]
- [-o](grants overwrite permission
- [-k](keeps voxel dimensions same as stardard file's--i.e.disables interpolation to cubic voxels)
- [-s intensity-scale-factor]
- [-x x-dim x-size [x-shift]]
- [-y y-dim y-size [y-shift]]
- [-z z-dim z-size [z-shift]]
- [-n model {x-half-window-width y-half-window-width z-half-window-width}]

where the following definitions apply:

`.air-file`

name of the input reslice parameter file

`output`

name of the output image file (.hdr or .img suffix optional)

`alternate-reslice-file`

name of an alternate reslice file that is spatially equivalent (same voxel sizes and dimensions) to the default reslice file specified in the .air-file

`intensity-scale-factor`

multiplicative intensity rescaling factor

`x-dim, y-dim, z-dim`

output file dimensions

x-size, y-size, z-size

output file voxel sizes

x-shift, y-shift, z-shift

shifts to apply to output file (in voxels)

model

one of the following should be selected:

- 0. nearest neighbor
- 1. trilinear
- 2. windowed sinc in original xy plane, linear along z
- 3. windowed sinc in original xz plane. linear along y
- 4. windowed sinc in original yz plane, linear along x
- 5. 3D windowed sinc
- 6. 3D windowed scanline sinc
- 7. 3D unwindowed scanline sinc
- 10. 3D scanline chirp-z
- 11. scanline chirp-z in original xy plane, linear along z

x-half-windown width, y-half-window width, z-half-window-width

The half-window widths which should only be supplied for windowed sinc interpolation models. Models 2-4 require two half-window widths. Models 5 and 6 require three half-window widths.

NOTE: The program interpolates output to create cubic voxels by default unless overridden by -k or -p option.

F.21.3.3 EXAMPLES:

reslice pet.air newpet

- if file newpet.img does not already exist, this command will reslice the reslice file identified in pet.air using the parameters in pet.air. The resulting file will be interpolated to cubic voxels (even if the standard file in pet.air did not have cubic voxels).

reslice pet.air newpet -o

- Same as above example but any existing file newpet.img will be overwritten.

reslice pet.air newpet -k -o

- Same as above example except that the output file voxels will have the same dimensions as the standard file (i.e. results will not be cubic voxels unless the standard file itself had cubic voxels).

reslice pet.air newpet -o -z 43 2.25

- The reslice file identified in pet.air will be resliced to generate 43 planes of output data with an interplane distance of 2.25, regardless of the standard file voxel z_size.

F.21.3.4 COMMENTS:

- Trilinear interpolation is the default interpolation method for this program.
- Interpolation models 6,7, 10 and 11 use scanline decomposition to accelerate interpolation. To minimize aliasing, scanline decomposition is preceeded by oversampling of the data in a prepass interpolation step. The AIR package is distributed using a conservative prepass interpolation that doubles the number of voxels along the necessary dimensions, a strategy that should be valid even with fairly large rotations (assuming cubic voxels and a rigid-body spatial transformation model). Prepass interpolation also increases memory requirements and resampling time, so it can be advantageous to reduce the amount of prepass interpolation. If you know that your rotations are always very small, you can adjust the amount of prepass interpolation in models 10 and 11 by reducing the values of BIGY and BIGZ in AIR3.0/src/chirpscan.c and AIR3.0/src/chirperxy.c and recompiling. Any value greater than 1.00 is acceptable. As a rough guide, you should allow for a prepass interpolation ($1/\cos^2(\text{theta})$) where theta is the largest expected rotation angle (this assumes cubic voxels). Models 6 and 7 unfortunately cannot be adjusted in this way. The issue of aliasing in scanline interpolation is complex and has not been fully addressed for 3D. For a discussion in 2D, see: Fraser, D, Schowengerdt RA. Avoidance of additional aliasing in multipass image rotations. IEEE Transactions on Image Processing 1994;3:721-735.
- If you are using the -z option to specify the number of planes and interplane distance, you may have computed these values to match your scanner's field of view. If you unexpectedly find that data for the last plane is always missing, this may be due to round-off errors. As an exception to the rule otherwise applied in the AIR package when rounding-off voxel sizes (general rule is round the smallest voxel size downward and all others upwards), here you may need to round the interplane distance downwards to avoid this problem.
- The half window widths for sinc interpolation control the number of surrounding voxels that contribute to the interpolated value along each axis of the reslice file. A half-window width of 1 results in interpolation along that axis that only includes the nearest neighbor on either side and a half-window width of 6 will include the six nearest neighboring voxels in either direction. The total number of voxel included for standard 3D sinc interpolation is therefore 8*(x_half_window_width)*(y_half_window_width*(z_half_window_width) . The larger the half windows,the more closely the interpolation will match true sinc interpolation, but also the slower the resampling process.
- Windowing of sinc interpolation is implemented using a Hanning window function exactly as described by Hajnal JV, Saeed N, Soar EJ, Oatridge A, Young IR, Bydder GM. Journal of Computer Assisted Tomography 1995:19:289-296.

F.21.3.5 ERROR MESSAGES: (ALPHABETICAL)

See also: Generic error messages

- File '___' not created because of matrix size incompatibility...

The reslice file matrix dimensions don't match those stored in the .air file.

Your .air file has gotten associated inappropriately with a different reslice file.

The programs scanair, and identify may help you sort it all out.

- File '___ not created because of voxel size discrepancy...

The reslice file voxel sizes don't match those stored in the .air file

Your .air file is inappropriately associated with an incorrect reslice file

The programs scanair, and identify may help you sort it all out.

F.21.4 reslice_warp

- purpose
- usage
- examples
- comments
- error messages
- see also

F.21.4.1 PURPOSE:

This is the program that takes .warp files and uses the information that they contain to load the corresponding image file and generate a new, realigned file.

F.21.4.2 USAGE:

reslice_warp .warp-file output [options]

options:

- [-a alternate-reslice-file]
- [-o] (grants overwrite permission)
- [-s intensity_scale_factor]
- [-n model {x-half-window-width y-half-window-width z-half-window-width}]

where the following definitions apply:

`.warp-file`

name of the .warp file containing the nonlinear transformation parameters

`output`

name of the output image file (.hdr or .img suffix optional)

`alternate-reslice-file`

name of an alternate reslice file that is spatially equivalent (same voxel sizes and dimensions) to the default reslice file specified in the .air-file

`intensity-scale-factor`

multiplicative intensity rescaling factor

`model`

one of the following should be selected:

- 0. nearest neighbor
- 1. trilinear
- 2. windowed sinc in original xy plane, linear along z
- 3. windowed sinc in original xz plane. linear along y
- 4. windowed sinc in original yz plane, linear along x
- 5. 3D windowed sinc

x-half-windown width, y-half-window width, z-half-window-width

The half-window widths which should only be supplied for windowed sinc interpolation models. Models 2-4 require two half-window widths. Model 5 requires three half-window widths.

NOTE: Unlike reslice, this program does not interpolate output to create cubic voxels.

F.21.4.3 EXAMPLES:

reslice_warp mri.warpatlas newmri -o

- The reslice file identified in mri.warpatlas will be resampled based on the nonlinear spatial transformation parameters in mri.warpatlas. The newly created file will be called newmri.img and will overwrite any existing file with that name.

F.21.4.4 COMMENTS:

- Trilinear interpolation is the default interpolation method for this program.
- The half window widths for sinc interpolation control the number of surrounding voxels that contribute to the interpolated value along each axis of the reslice file. A half-window width of 1 results in interpolation along that axis that only includes the nearest neighbor on either side and a half-window width of 6 will include the six nearest neighboring voxels in either direction. The total number of voxel included for standard 3D sinc interpolation is therefore 8*(x_half_window_width)*(y_half_window_width*(z_half_window_width) . The larger the half windows,the more closely the interpolation will match true sinc interpolation, but also the slower the resampling process.
- If you have registered data using edited versions of files but then reslice unedited versions, you may see bizarre reduplications of data outside the brain. This is due to the fact that the nonlinear transformation can cause the reslice file to bend back on itself or create mirror image ghosts. In theory this

can even happen within the brain, but you should not typically encounter this unless you have ignored advice about use of the program align_warp.

- Windowing of sinc interpolation is implemented using a Hanning window function exactly as described by Hajnal JV, Saeed N, Soar EJ, Oatridge A, Young IR, Bydder GM. Journal of Computer Assisted Tomography 1995:19:289-296.

F.21.4.5 ERROR MESSAGES: (ALPHABETICAL BY CASE)

See also: Generic error messages

- A positive number must follow -s

self-explanatory

- A valid interpolation model number must follow -n

provide a number from the menu

- Argument that follows -s cannot start with - ; it must be a positive number

self-explanatory

- File '____' not created because of matrix size incompatibility

The reslice file matrix dimensions don't match those stored in the .air file.

Your .air file has gotten associated inappropriately with a different reslice file.

The programs scan_warp, and identify may help you sort it all out.

- File '____' not created because of voxel size discrepancy

The reslice file voxel sizes don't match those stored in the .air file

Your .air file is inappropriately associated with an incorrect reslice file

The programs scan_warp, and identify may help you sort it all out.

- Interpolation windows cannot be negative

self explanatory

- Sorry, flag -____ is not defined for this program

self explanatory, note that this program does not include all of the options found in reslice

F.22 AIR FILE TYPES

The following types of files are utilized in the AIR package:

- header (.hdr) files
  - type 0
  - type 1
  - type 2
  - type 3
- image (.img) files
- registration parameter .air files
- registration parameter .warp files
- initialization (.init) files

F.22.1 Header (.hdr) files:

The AIR package utilizes separate header and image files. The header file contains all of the information necessary to interpret the data in the image file. At present, the default header format for the AIR package is compatible with the ANALYZE header format from the Mayo Clinic. All default header files must end with the suffix .hdr and a corresponding image file with the suffix .img is expected to exist. The image files contain pixel values without any associated header information. If you can get your data into raw format (i.e., data only, without any header information), the AIR package provides all of the utilities needed to create the corresponding header files.

The following pieces of information are stored in the header files and are used by the AIR package:

- the number of bits per pixel in the image (8 bit and 16 bit images are supported)
- the image matrix x-dimension
- the image matrix y-dimension
- the image matrix z-dimension
- the image voxel x-size
- the image voxel y-size
- the image voxel z-size
- the global maximum for the image
- the global minimum for the image

The units for the image voxel sizes are not specified in the AIR package (millimeters are recommended), and **identical units must be used for all three dimensions.**

If voxel sizes have to be rounded-off, the smallest voxel size should be rounded downwards and the other voxel sizes rounded upwards to assure that interpolation to cubic voxels will generate the correct number of planes.

The default header currently contains space to store additional information, but storage and maintenance of such information is not supported by the AIR package.

A C programmer can modify the air package to read and write other data file types, including files that combine header and image information into a single file, but this is a significant undertaking.

You can create a header file using makeaheader, review the information in a header file using scanheader, modify the header voxel sizes with fixheader, and adjust the header global maximum with setheadermax.

When creating a new header, you must provide all of the information that will be contained in the header. In order to properly specify the image matrix dimensions, the voxel dimensions, and the number of bits per pixel, you must know how data is represented in your image file. There are two major considerations:

The x, y, and z dimensions must be defined according to how the data is ordered in the image file. Your image display package may define the dimensions differently.

For 16 bit data, you must know what 16 bit variable type was used to store the data and what 16 bit numerical value is supposed to represent a "black" pixel.

F.22.2 Image (.img) files:

The AIR package utilizes separate header and image files. Image files end with the suffix '.img'. A single image file contains all of the data for the entire three dimensional volume stored row after row, plane after plane. The image file consists of "raw" voxel intensity values (8 bit and 16 bit images are supported) that are stored sequentially. No other information is contained in the .img file.

The image file's voxel order is defined as follows:

> The file *x-dimension* is defined as the dimension that changes most rapidly (i.e., each sequential voxel will fall in a different column and will therefore have a different x coordinate).
>
> The file *y-dimension* changes more slowly than the x-dimension and more quickly than the z-dimension.
>
> The file *z-dimension* is defined as the dimension that changes most slowly (e.g., all of the voxels for a given z-plane are stored before any of the voxels of the next z-plane).

Please note that your image display package may define the dimensions differently. In addition, note that the internal coordinate system used for indexing the voxels once they are loaded into the AIR package may differ substantially from the coordinate system used by your image display package.

Many programs in the AIR package will generate new image files.

F.22.3 Registration parameter .air files

Linear spatial transformations in AIR are stored in the form of .air files.

.air files are created and used by the AIR package and contain the following pieces of information:

The "reslice" file
the name of the image file to be spatially transformed by the .air file

Definition of a "standard" space
the matrix size and voxel dimensions that will result after transforming the reslice file.

A transformation matrix
a description, in linear algebraic terms, of the spatial transformation to be applied

Additional, nonessential information
information about how the .air file was generated.

The contents of a .air file can be display using scanair.

A number of programs in the AIR package will create or modify .air files.

F.22.4 Registration parameter .warp files

Nonlinear spatial transformations in AIR are stored in the form of .warp files.

.warp files are created and used by the AIR package and contain the following pieces of information:

The "reslice" file
the name of the image file to be spatially transformed by the .air file

Definition of a "standard" space
the matrix size and voxel dimensions that will result after transforming the reslice file.

A set of transformation parameters.
a set of equations that describe the spatial transformation to be applied

Additional, nonessential information
information about how the .warp file was generated.

The contents of a .warp file can be display using scan_warp.

A number of programs in the AIR package will create or modify .warp files.
Initialization (.init) files:

Initialization files are used to override the default center-of-file to center-of-file, no rotation or rescaling initialization used by the automated alignment programs. These are ASCII files and their contents may vary for different automated alignment programs. For rigid-body, global rescaling, and traditional 9 parameter Talairach models, initialization files can be created by manualreslice. The format of initialization files for other models are described with each model.

F.23 AIR INTERNAL COORDINATE SYSTEM

The internal coordinate system used by the AIR package will not necessarily assign the same axes or voxel coordinates that your image display package assigns. The internal coordinate axes of an image file are defined by the order in which the image voxels are stored in the file with the x-dimension changing most often (with every voxel) and the z-dimension least often. When an image file is loaded, the first voxel in the file is assigned the internal (x,y,z) coordinates (0,0,0). Assuming that the x-dimension of the image matrix is greater than 1, the second voxel in the file is assigned the internal coordinates (1,0,0). After a number of pixels equal to the file x-dimension (as specified in the corresponding header file) has been loaded, the next voxel is assigned the internal coordinates (0,1,0),etc. The last voxel in the file is assigned coordinates (x-dimension -1,y-dimension -1,z-dimension -1). The legal range for each coordinate is as follows:

- x-coordinate: 0 to x-dimension -1
- y-coordinate: 0 to y-dimension -1
- z-coordinate: 0 to z-dimension -1

The internal coordinate system is the one referenced by the homogenous transformation matrix contained in the .air files.

Your image display package may assign different coordinates to voxels for a number of reasons:

- Legal coordinates may start with the number 1 instead of the number 0.
- The x, y, and z axes may be interchanged.
- The numbering order along one or more axes may be reversed.

In general, the AIR package does not allow users to input coordinate locations, so understanding of the internal coordinate system is not critical for routine usage. The user does need to know the definitions of the coordinate axes to create initialization files and to use the program manualreslice. If documentation about your display package is not available, you can determine these definitions empirically by using manualreslice to shift an image along one axis and then comparing the images before and after shifting using your image display package. This trial and error approach can also be used to determine whether to use positive or negative values to achieve a desired effect.

F.24 VOXEL ANISOTROPY AND INTERPOLATION TO CUBIC VOXELS

- voxel size anisotropy
- interpolation to cubic voxels
- unexpected post-interpolation dimensions
- caution about manually defined interpolation

F.24.1 Voxel size anisotropy

AIR 3.0 allows voxel sizes to be anisotropic in all three directions. The real world voxel sizes are stored in the header files and can be displayed with

scanheader. If the values are incorrect, they can be modified with fixheader. **The voxel dimensions in the header must always be the actual, objectively determined dimensions.** Incorrect specification of voxel dimensions will invalidate many of the spatial transformation models used in the AIR package.

You should adopt a standard unit of measure for the header file voxel sizes (millimeters is recommended) and apply the standard without exception. It is critical to the mathematical models in the AIR package that all voxel sizes be expressed in the same units.

F.24.2 Interpolation to cubic voxels

Some of the programs in the AIR package have defaults or options to interpolate output files to cubic voxels. Indeed, the matrix stored in .air files specifies a transformation that will generate cubic voxels. The smallest of the three target voxel sizes (x size, y size or z size) becomes the voxel size of the interpolated volume. The origin of the internal coordinate system is used as the origin for the interpolation. The interpolation homogenous coordinate transformation matrix is:

$$\begin{bmatrix} sxoom & 0 & 0 & 0 \\ 0 & syoom & 0 & 0 \\ 0 & 0 & szoom & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

sxoom=(standard file voxel x size) / (smallest standard file voxel size)
syoom=(standard file voxel y size) / (smallest standard file voxel size)
szoom=(standard file voxel z size) / (smallest standard file voxel size)

smallest standard file voxel size=min(standard file voxel x, y and z sizes)

Using this matrix to define the interpolation function, the location of the first voxel in the file (x=y=z=0) is unaltered by the transformation. The AIR package will only interpolate within the boundaries of the uniterpolated data, it will not extrapolate outside these boundaries. The total number of interpolated points along a given dimension that can be generated without extrapolating beyond the boundaries of the original data is given by the equation:

newdim=int((vox / ssize)*(dim-1)+1)

where:

- dim is the dimension of interest before interpolation
- newdim is the dimension of interest after interpolation
- vox is the voxel size of the file along the dimension of interest before interpolation
- ssize is the smallest standard file voxel size
- int() is the integer function (the non-integer portion is truncated without rounding)

F.24.3 Unexpected post-interpolation dimensions

The fact that the AIR package will not extrapolate outside the boundaries of the original uninterpolated data can result in post-interpolations file dimensions that are different from those that you might expect. For example, if your file has a voxel x size of 2.5 and a voxel y size of 1.25, interpolation will lead to an x dimension that is twice the original x dimension **minus one** (assuming that the z dimension is greater than or equal to 1.0).

Roundoff errors can also lead to unexpected dimensions after interpolation. If you need to round-off voxel sizes, you should round the smallest voxel size downwards and all other voxels sizes upwards to prevent loss of an entire plane of data from round-off errors during interpolation.

If you are working with two dimensional data, be sure to specify a voxel z size that is greater than or equal to the voxel x and y sizes. Otherwise, interpolation to the irrelevant voxel z size will occur in order to generate cubic voxels.

Other registration and display packages that interpolate data to cubic voxels may define the interpolation function differently (e.g., they may use the center of the file as the interpolation origin and consequently generate one less interpolated plane). The post-interpolation coordinate assigned to a given real world location can vary by as much as 1 voxel as a result of differing definitions of the interpolation transformation. If the center of the file is used as the interpolation origin by other registration and display packages, incorrect rounding of voxel sizes can lead to interpolation trucation of two planes of original data when the rounding conventions described above are not followed.

F.24.4 Caution about manually defined interpolation

It is possible to use the program manualreslice to interpolate a file to cubic voxels or to specify a .air file to perform this transformation by specifying rotations and translations of zero and scaling factors of 1.0. Please be aware that manualreslice uses a center-of-file-as-origin model by default and consequently may not produce the same result that would have been obtained using the automatic interpolation function used elsewhere in the AIR package. To circumvent this problem, manualreslice will offer you a top-of-file-as-origin model when certain criteria suggesting that you are trying to define an interpolation function are met. If you are offered this option and apply it, the result will be identical to the automatic interpolation described on this page.

F.25 .AIR FILE HOMOGENOUS COORDINATE TRANSFORMATION MATRIX

F.25.1 Definition

By definition, the homogenous coordinate transformation matrix stored in a .air file should transform internal voxel coordinate locations in an interpolated (i.e.,interpolated to cubic voxels) version of the standard file to the corresponding uninterpolated (i.e., not necessarily cubic voxels) internal voxel coordinate locations in the reslice file. By convention, the last element of the matrix should be converted to a one by dividing all elements by the last element.

Mathematically, these specifications imply that:

$$\begin{bmatrix} T*x' \\ T*y' \\ T*z' \\ T \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & m \\ n & p & q & 1 \end{bmatrix} * \begin{bmatrix} sxoom & 0 & 0 & 0 \\ 0 & syoom & 0 & 0 \\ 0 & 0 & szoom & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where:

x, y and z represent standard file internal voxel coordinates (not interpolated to cubic voxels)
x',y' and z' represent reslice file internal voxel coordinates (not interpolated to cubic voxels)

sxoom=(standard file voxel x size) / (smallest standard file voxel size)
syoom=(standard file voxel y size) / (smallest standard file voxel size)
szoom=(standard file voxel z size) / (smallest standard file voxel size)

smallest standard file voxel size=min(standard file voxel x, y and z sizes)

T is a constant.

and

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & m \\ n & p & q & 1 \end{bmatrix}$$

is the matrix that is stored in the .air file.

F.25.2 Displaying the native .air file matrix

The .air file matrix can be displayed using the program scanair with the default display mode.

In the C code, the .air file transformation matrix is stored in the array 'e'. In version 2.0 of the AIR package, the elements of this array have an orderly relationship to the .air file homogenous coordinate transformation matrix:

$$\begin{bmatrix} e[0][0] & e[1][0] & e[2][0] & e[3][0] \\ e[0][1] & e[1][1] & e[2][1] & e[3][1] \\ e[0][2] & e[1][2] & e[2][2] & e[3][2] \\ e[0][3] & e[1][3] & e[2][3] & e[3][3] \end{bmatrix}$$

In version 1.0 of the AIR package, a different format was used.

F.25.3 Displaying modified versions of the .air file matrix

If you need a matrix that will find the voxel in the (uninterpolated) reslice file that corresponds to a voxel in the original uninterpolated version of the standard file, you can display such a matrix using the program scanair with the -v flag.

To modify the C array 'e' to display this modified matrix:

- multiply e[0][0],e[0][1],e[0][2] and e[0][3] by sxoom
- multiply e[1][0],e[1][1],e[1][2] and e[1][3] by syoom
- multiply e[2][0],e[2][1],e[2][2] and e[2][3] by szoom

(Note that sxoom, syoom and szoom are defined above).

If you need a matrix in real world units, you can display such a matrix using the program scanair with the -r flag.

To modify the C array 'e' to display this modified matrix:

- divide e[0][0], e[0][1], e[0][2] and e[0][3] by the smallest standard file voxel size
- divide e[1][0], e[1][1], e[1][2] and e[1][3] by the smallest standard file voxel size
- divide e[2][0], e[2][1], e[2][2] and e[2][3] by the smallest standard file voxel size
- multiply e[0][0], e[1][0], e[2][0] and e[3][0] by the reslice file voxel x size
- multiply e[0][1], e[1][1], e[2][1] and e[3][1] by the reslice file voxel y size
- multiply e[0][2], e[1][2], e[2][2] and e[3][2] by the reslice file voxel z size

If you are using a rigid-body model, the resulting matrix should specify an orthonormal transformation (you should ignore the last row and last column of the 4x4 matrix when testing for orthonormality). Note that the origin referenced by this matrix is at the same physical location as the origin of the internal voxel coordinate system used by AIR.

F.25.4 Upgrading from AIR 1.0

All AIR 3.0 programs are compatible with .air files generated by AIR 1.0 (the reverse is not true). The AIR 3.0 subroutine that loads .air files uses the size of a given .air file to determine which format is appropriate and coverts AIR 1.0 data into the new format transparently. The AIR 3.0 implementation of .air files is identical to AIR 2.0.

In version 1.0 of the AIR package, the array 'e' did not have storage for the last row of the homologous coordinate transformation matrix and this row was assigned fixed values of 0,0,0 and 1. In addition, the columns of the homologous coordinate transformation matrix were assigned in a different, somewhat disorderly fashion:

$$\begin{bmatrix} e[1][0] & e[2][0] & e[3][0] & e[0][0] \\ e[1][1] & e[2][1] & e[3][1] & e[0][1] \\ e[1][2] & e[2][2] & e[3][2] & e[0][2] \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

AIR 1.0 required that pixels be square (i.e., that voxel x and y sizes be identical), and assumed that the voxel z size was greater than or equal to the voxel x and y size. Consequently sxoom and syoom were both equal to 1 in AIR 1.0.

Note that perspective transformations could not be represented in version 1.0 of the AIR package.

F.26 .WARP FILE TRANSFORMATIONS

In AIR, .warp files are the nonlinear equivalents of .air files. However, the specification of .warp files differ in several ways. First of all, the transformation specified in a .warp file will convert internal voxel coordinate locations in an **uninterpolated** version of the standard file to the corresponding uninterpolated internal voxel coordinate locations in the reslice file. Homogenous coordinates are not used (the .warp file does include storage for possible future implementation of homogenous coordinates). All transformations are stored as fifth order polynomials, starting with the lowest order terms (the pure translations) and ending with the highest order terms (this is in contrast to.air files which the reverse priority). If the letters x, y, and z within a term are placed in reverse alphabetic order, the terms of the same order are prioritized alphabetically (e.g., for third order terms the priority is xxx yxx yyx yyy zxx zyx zyy zzx zzy zzz).

The .warp file transformation can be displayed using the program scan_warp.

For a voxel with coordinates (x,y,z) in the standard file, the corresponding coordinates (x',y',z') in the reslice file are computed in C code using the array e as:

x'=e[0][0]+e[1][0]*x+e[2][0]*y+e[3][0]*z+e[4][0]*xx +...+e[10][0]*xxx + ... + e[20][0]*xxxx+ ...+e[35][0]*xxxxx+...+e[55][0]*zzzzz

y'==e[0][1]+e[1][1]*x+e[2][1]*y+e[3][1]*z+e[4][1]*xx +...+e[10][1]*xxx + ... + e[20][1]*xxxx+ ...+e[35][1]*xxxxx+...+e[55][1]*zzzzz

z' =e[0][2]+e[1][2]*x+e[2][2]*y+e[3][2]*z+e[4][2]*xx +...+e[10][2]*xxx + ... + e[20][2]*xxxx+ ...+e[35][2]*xxxxx+...+e[55][2]*zzzzz

F.26.1 AIR nonlinear tranformations - .warp File Transformations

In AIR, .warp files are the nonlinear equivalents of .air files. However, the specification of .warp files differ in several ways. First of all, the transformation specified in a .warp file will convert internal voxel coordinate locations in an **uninterpolated** version of the standard file to the corresponding uninterpolated internal voxel coordinate locations in the reslice file. Homogenous coordinates are not used (the .warp file does include storage for possible future implementation of homogenous coordinates). All transformations are stored as fifth order polynomials, starting with the lowest order terms (the pure translations) and ending with the highest order terms (this is in contrast to.air files which the reverse priority). If the letters x, y, and z within a term are placed in reverse alphabetic order, the terms of the same order are prioritized alphabetically (e.g., for third order terms the priority is xxx yxx yyx yyy zxx zyx zyy zzx zzy zzz).

The .warp file transformation can be displayed using the program scan_warp.

For a voxel with coordinates (x,y,z) in the standard file, the corresponding coordinates (x',y',z') in the reslice file are computed in C code using the array e as:

x'=e[0][0]+e[1][0]*x+e[2][0]*y+e[3][0]*z+e[4][0]*xx +...+e[10][0]*xxx + ... + e[20][0]*xxxx+ ...+e[35][0]*xxxxx+...+e[55][0]*zzzzz

y'==e[0][1]+e[1][1]*x+e[2][1]*y+e[3][1]*z+e[4][1]*xx +...+e[10][1]*xxx + ... + e[20][1]*xxxx+ ...+e[35][1]*xxxxx+...+e[55][1]*zzzzz

z' =e[0][2]+e[1][2]*x+e[2][2]*y+e[3][2]*z+e[4][2]*xx +...+e[10][2]*xxx + ... + e[20][2]*xxxx+ ...+e[35][2]*xxxxx+...+e[55][2]*zzzzz

F.27 SPATIAL TRANSFORMATION MODELS

The AIR package uses a variety of spatial models to restrict the type of linear transformation that is being applied when performing registration. While the most general model could be used in all instances, to do so would be both unnecessarily slow and in fact, inaccurate. For example, when if voxel sizes are known exactly, it makes little sense to use anything other than a rigid-body model for intrasubject registration unless there are strong reasons to suspect that the shape and/or size of the brain has changed between image acquisitions. In this context, any values for the nine extra parameters that imply anything other than an exact rigid-body fit might well be considered erroneous.

F.28 THE FOLLOWING 3D LINEAR MODELS HAVE BEEN IMPLEMENTED IN AIR 3.0:

F.28.1 Rigid Body Transformations

- transformation matrix
  - standard file interpolation matrix
  - standard file centering matrix
  - pixel size correction matrix
  - rigid body rotation matrix
  - rigid body translation matrix
  - reslice file inverse centering matrix
  - reslice file inverse interpolation matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages

Programs that incorporate the rigid body transformation model:

- alignlinear
- manualreslice
- alignpettopet (AIR 1.0)
- alignmritopet (AIR 1.0)
- alignpettomri (AIR 1.0)

F.28.1.1 TRANSFORMATION MATRIX

The rigid body model requires that the real world Euclidean distance between any two coordinate locations to remain unchanged by the transformation. Since the AIR package allows anisotropic voxels sizes within a given file as well as different voxel sizes between files, these factors must be taken into account when applying a rigid body transformation. In the AIR package, the rigid body model is parameterized in terms of rotations around and translations along each of the three major coordinate axes. In order to make these parameters more inituitive, the rotations of the rigid body transformation are defined as taking place around the centers of the files rather than the origin of the internal coordinate system (located at one corner of the file).

The rigid body transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as the product of a series of homogenous transformation matrices:

(reslice file internal coordinates)=Zr*Cr*T*R*P*Cs*Zs*(standard file internal coordinates)

where

- Zs corrects for voxel size anisotropy in the standard file and is omitted when the reslice file is to be resampled to generate cubic voxels.
- Cs shifts the coordinate system to the center of the standard file
- P corrects for differences in pixel size in the two files
- R performs a rigid body rotation
- T performs a rigid body translation
- Cr shifts the coordinate system from the center back to one corner of the reslice file
- Zr corrects for voxel size anisotropy in the reslice file

F.28.1.1.1 Standard file interpolation matrix

Zs =

$$\begin{bmatrix} sxoom & 0 & 0 & 0 \\ 0 & syoom & 0 & 0 \\ 0 & 0 & szoom & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

sxoom=(standard file voxel x size) / (smallest standard file voxel size)
syoom=(standard file voxel y size) / (smallest standard file voxel size)
szoom=(standard file voxel z size) / (smallest standard file voxel size)

smallest standard file voxel size=min(standard file voxel x, y and z sizes)

This homogenous transformation matrix remaps coordinate locations in the standard file to new coordinates with cubic voxels. The origin remains at (0,0,0).

F.28.1.1.2 Standard file centering matrix

Cs=

$$\begin{bmatrix} 1 & 0 & 0 & -(sx\_dim-1)*sxoom/2 \\ 0 & 1 & 0 & -(sy\_dim-1)*syoom/2 \\ 0 & 0 & 1 & -(sz\_dim-1)*szoom/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

sx_dim=standard file x-dimension
sy_dim=standard file y-dimension
sz_dim=standard file z-dimension
sxoom=(standard file voxel x size) / (smallest standard file voxel size)
syoom=(standard file voxel y size) / (smallest standard file voxel size)
szoom=(standard file voxel z size) / (smallest standard file voxel size)

smallest standard file voxel size=min(standard file voxel x, y and z sizes)

This homogenous coordinate transformation matrix shifts the origin from (0,0,0) to the exact center of the standard file.

F.28.1.1.3 Pixel size correction matrix

P =

$$\begin{bmatrix} ssize/rsize & 0 & 0 & 0 \\ 0 & ssize/rsize & 0 & 0 \\ 0 & 0 & ssize/rsize & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

ssize=min(standard file voxel x, y and z sizes)
rsize=min(reslice file voxel x, y and z sizes)

Coordinate units are modified by this homogenous coordinate transformation matrix to be equivalent to those used in an interpolated version of the reslice file.

F.28.1.1.4 Rigid body rotation matrix

R =

$$\begin{bmatrix} (\cos\varphi*\cos\phi+\sin\varphi*\sin\theta*\sin\phi) & (\sin\varphi*\cos\phi-\cos\varphi*\sin\theta*\sin\phi) & (\cos\theta*\sin\phi) & 0 \\ (-\sin\varphi*\cos\theta) & (\cos\varphi*\cos\theta) & (\sin\theta) & 0 \\ (\sin\varphi*\sin\theta*\cos\phi-\cos\varphi*\sin\phi) & (-\cos\varphi*\sin\theta*\cos\phi-\sin\varphi*\sin\phi) & (\cos\theta*\cos\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:
θ=rotation around the x-axis (pitch)
ϕ=rotation around the y-axis (roll)
φ=rotation around the z-axis (yaw)

This homogenous coordinate transformation matrix performs rotations while preserving Euclidean distances between coordinate locations.

F.28.1.1.5 Rigid body translation matrix

T =

$$\begin{bmatrix} 1 & 0 & 0 & \text{x-shift} \\ 0 & 1 & 0 & \text{y-shift} \\ 0 & 0 & 1 & \text{z-shift} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:
x-shift=translation along the x-axis
y-shift=translation along the y-axis
z-shift=translation along the z-axis

This homogenous coordinate transformation matrix performs translations while preserving Euclidean distances between coordinate locations.

Note that the shifts have units of interpolated reslice file voxels

F.28.1.1.6 Reslice file inverse centering matrix

Cr =

$$\begin{bmatrix} 1 & 0 & 0 & (\text{rx\_dim}-1)*\text{rxoom}/2 \\ 0 & 1 & 0 & (\text{ry\_dim}-1)*\text{ryoom}/2 \\ 0 & 0 & 1 & (\text{rz\_dim}-1)*\text{rzoom}/2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

rx_dim=reslice file x-dimension
ry_dim=reslice file y-dimension
rz_dim=reslice file z-dimension
rxoom=(reslice file voxel x size) / (smallest reslice file voxel size)
ryoom=(reslice file voxel y size) / (smallest reslice file voxel size)
rzoom=(reslice file voxel z size) / (smallest reslice file voxel size)

smallest reslice file voxel size=min(reslice file voxel x, y and z sizes)

The origin of the coordinate system is shifted from the center of the file to internal coordinate (0,0,0) of the reslice file by this homogenous coordinate transformation matrix.

F.28.1.1.7 Reslice file inverse interpolation matrix

Zr =

$$\begin{bmatrix} 1/rxoom & 0 & 0 & 0 \\ 0 & 1/ryoom & 0 & 0 \\ 0 & 0 & 1/rzoom & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

rxoom=(reslice file voxel x size) / (smallest reslice file voxel size)
ryoom=(reslice file voxel y size) / (smallest reslice file voxel size)
rzoom=(reslice file voxel z size) / (smallest reslice file voxel size)

smallest reslice file voxel size=min(reslice file voxel x, y and z sizes)

Cubic voxel coordinate locations are remapped to the actual voxel locations in the reslice file by this homogenous coordinate transformation matrix. The origin (0,0,0) remains unchanged.

F.28.1.2 REPRESENTATION IN .AIR FILES

Since the standard file interpolation matrix Zs is already implicit in the definition of the.air file transformation matrix, it is omitted from the matrix defined above when creating the .air file matrix: Zr*Cr*T*R*P*Cs.

F.28.1.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the rigid body model consist of the following in ASCII format:

- pitch (in radians)
- roll (in radians)
- yaw (in radians)
- 2*x_shift (x_shift in units of reslice file voxels)
- 2*y_shift (y_shift in units of reslice file voxels)
- 2*z_shift (z_shift in units of reslice file voxels)

The reslice file voxel units referred to above are cubic (i.e., already interpolated to correct for reslice file voxel size anisotropy).

F.28.1.4 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using a rigid body model is:

- pitch=0
- roll=0
- yaw=0

- x_shift=0
- y_shift=0
- z_shift=0

This results in the exact centers of the two files being aligned to one another.

F.28.1.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define a rigid body transformation. Without explicit equations such as those provided above, the terms pitch, roll, yaw, x-shift, y-shift, and z-shift are ambiguous. Other packages for registering or displaying images may not apply the transformations in the same order as the AIR package, so simple substitution of nominal parameters called "pitch","roll", etc. from other packages may not produce the desired result. In moving from standard file coordinates to reslice file coordinates, the AIR package performs rotations around the z-axis (yaw), x-axis (pitch), and y-axis (roll) in that order followed by translations x-shift, y-shift, and z-shift. Rotations operate around the exact centers of the files (before and after interpolation) and the sign conventions for both rotations and translations are arbitrary and may differ from those used in your alternative package. Shifts are defined in a version of reslice file space that has been interpolated to cubic voxels and are expressed in units of cubified voxels.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.28.2 Global Rescaling Transformations (7 parameters)

- transformation matrix
  - standard file interpolation matrix
  - standard file centering matrix
  - pixel size correction matrix
  - rescaling matrix
  - rigid body rotation matrix
  - rigid body translation matrix
  - reslice file inverse centering matrix
  - reslice file inverse interpolation matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages
- Programs that incorporate the global rescaling transformation model:
  - alignlinear

F.28.2.1 TRANSFORMATION MATRIX

The global rescaling model requires that the real world Euclidean distance between any two coordinate locations to by multiplied by a constant as a result of the transformation. Since the AIR package allows anisotropic voxels sizes within a given file as well as different voxel sizes between files, these factors must be taken into account when applying a global rescaline transformation. In the AIR package, the global rescaling model is parameterized in terms of rotations around and translations along each of the three major coordinate axes and a global rescaling term. In order to make these parameters more inituitive, the rotations of the global rescaling transformation are defined as taking place around the centers of the files rather than the origin of the internal coordinate system (located at one corner of the file).

The global rescaling transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as the product of a series of homogenous transformation matrices:

(reslice file internal coordinates)=Zr*Cr*T*R*G*P*Cs*Zs*(standard file internal coordinates)

where

- Zs corrects for voxel size anisotropy in the standard file and is omitted when the reslice file is to be resampled to generate cubic voxels.
- Cs shifts the coordinate system to the center of the standard file
- P corrects for differences in pixel size in the two files
- G performs global rescaling
- R performs a rigid body rotation
- T performs a rigid body translation
- Cr shifts the coordinate system from the center back to one corner of the reslice file
- Zr corrects for voxel size anisotropy in the reslice file

F.28.2.1.1 Standard file interpolation matrix

See F.28.1.1.1 Standard file interpolation matrix above

F.28.2.1.2 Standard file centering matrix

See F.28.1.1.2 Standard file centering matrix above

F.28.2.1.3 Pixel size correction matrix

See F.28.1.1.3 Pixel size correction matrix above

F.28.2.1.4 Global rescaling matrix

$$G=\begin{bmatrix} scale & 0 & 0 & 0 \\ 0 & scale & 0 & 0 \\ 0 & 0 & scale & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This homogenous coordinate transformation matrix performs global rescaling.

F.28.2.1.5 Rigid body rotation matrix

See F.28.1.1.4 Rigid body rotation matrix above

F.28.2.1.6 Rigid body translation matrix

See F.28.1.1.5 Rigid body translation matrix above

F.28.2.1.7 Reslice file inverse centering matrix

See F.28.1.1.6 Reslice file inverse centering matrix above

F.28.2.1.8 Reslice file inverse interpolation matrix

See F.28.1.1.7 Reslice file inverse interpolation matrix above

F.28.2.2 REPRESENTATION IN .AIR FILES

See F.28.1.2 Representation in .air files above

F.28.2.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the global rescaling model consist of the following in ASCII format:

- pitch (in radians)
- roll (in radians)
- yaw (in radians)
- 2*x_shift (x_shift in units of reslice file cubified voxels)
- 2*y_shift (y_shift in units of reslice file cubified voxels)
- 2*z_shift (z_shift in units of reslice file cubified voxels)
- scale

The reslice file voxel units referred to above are cubic (i.e., already interpolated to correct for reslice file voxel size anisotropy).

F.28.2.4 DEFAULT INITIALIZATION

See F.28.1.4 Default initialization above

F.28.2.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define a global rescaling transformation. Without explicit equations such as those provided above, the terms scale, pitch, roll, yaw, x-shift, y-shift, and z-shift are ambiguous. Other packages for registering or displaying images may not apply the transformations in the same order as the AIR package, so simple substitution of nominal parameters called "pitch","roll", etc. from other packages may not produce the desired result. In moving from standard file coordinates to reslice file coordinates, the AIR package performs scaling, rotations around the z-axis (yaw), x-axis (pitch), and y-axis (roll) in that order followed by translations x-shift, y-shift, and z-shift. Rotations operate around the exact centers of the files (before and after interpolation) and the sign conventions for both rotations and translations are arbitrary and may differ from those used in your alternative package. Shifts are

defined in a version of reslice file space that has been interpolated to cubic voxels and are expressed in units of cubified voxels.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.28.3 Traditional 9 Parameter Transformations

- transformation matrix
  - standard file interpolation matrix
  - standard file centering matrix
  - pixel size correction matrix
  - rescaling matrix
  - rigid body rotation matrix
  - rigid body translation matrix
  - reslice file inverse centering matrix
  - reslice file inverse interpolation matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages
- Programs that incorporate the traditional 9 parameter transformation model:
  - alignlinear
  - manualreslice

F.28.3.1 TRANSFORMATION MATRIX

The traditional 9 parameter model performs rigid-body rotations and translations, followed by independent rescaling along the resultant x, y, and axes. Many would refer to this as the "Talairach model", but I have avoided this terminology because 1) Talairach, et.al, actually used a 13 parameter model that combined 12 such transformations in a piecemeal fashion, 2) much (or even most) of the published data alleged to be in "Talairach space" was transformed into that space using nonlinear transformations, not the traditional 9 parameter transformation described here, 3) unless the target to which you are registering has been properly oriented with respect to the AC-PC line, etc., this model will not allow you to report Talairach coordinates, and 4) this is not the optimum linear model in the AIR package to use for intersubject registration and I don't want to implicitly endorse this model as the model to use to derive Talairach coordinates. My general advice is to use the affine model to perform linear registration to a Talairach target or else to procede to a nonlinear technique. I reserve the traditional 9 parameter model for situations where I have reason to believe that rescaling truly should be applied along a certain set of axes (e.g., when I think

that the standard file voxel dimensions that I am using might be incorrect, or when they are unknown).

Since the AIR package allows anisotropic voxels sizes within a given file as well as different voxel sizes between files, these factors must be taken into account when applying a traditional 9 parameter transformation. In the AIR package, the traditional 9 parameter model is parameterized in terms of rotations around and translations along each of the three major coordinate axes and independent rescaling terms along each of the three major axes of the standard file. In order to make these parameters more inituitive, the rotations of the transformation are defined as taking place around the centers of the files rather than the origin of the internal coordinate system (located at one corner of the file).

The traditional 9 parameter transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as the product of a series of homogenous transformation matrices:

(reslice file internal coordinates)=Zr*Cr*T*R*G*P*Cs*Zs*(standard file internal coordinates)

where

- Zs corrects for voxel size anisotropy in the standard file and is omitted when the reslice file is to be resampled to generate cubic voxels.
- Cs shifts the coordinate system to the center of the standard file
- P corrects for differences in pixel size in the two files
- S performs rescaling along the standard file major axes
- R performs a rigid body rotation
- T performs a rigid body translation
- Cr shifts the coordinate system from the center back to one corner of the reslice file
- Zr corrects for voxel size anisotropy in the reslice file

F.28.3.1.1 Standard file interpolation matrix

See F.28.1.1.1 Standard file interpolation matrix above

F.28.3.1.2 Standard file centering matrix

See F.28.1.1.2 Standard file centering matrix above

F.28.3.1.3 Pixel size correction matrix

See F.28.1.1.3 Pixel size correction matrix above

F.28.3.1.4 Rescaling matrix

$S =$

$$\begin{bmatrix} xscale & 0 & 0 & 0 \\ 0 & yscale & 0 & 0 \\ 0 & 0 & zscale & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

xscale=rescaling along standard file x dimension
yscale=rescaling along standard file y dimension
zscale=rescaling along standard file z dimension

This homogenous coordinate transformation matrix rescales independently along the major axes of the standard file.

F.28.3.1.5 Rigid body rotation matrix

See F.28.1.1.4 Rigid body rotation matrix above

F.28.3.1.6 Rigid body translation matrix

See F.28.1.1.5 Rigid body translation matrix above

F.28.3.1.7 Reslice file inverse centering matrix

See F.28.1.1.6 Reslice file inverse centering matrix above

F.28.3.1.8 Reslice file inverse interpolation matrix

See F.28.1.1.7 Reslice file inverse interpolation matrix above

F.28.3.2 REPRESENTATION IN .AIR FILES

See F.28.1.2 Representation in .air files above

F.28.3.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the traditional 9 parameter model consist of the following in ASCII format:

- pitch (in radians)
- roll (in radians)
- yaw (in radians)
- 2*x_shift (x_shift in units of reslice file cubified voxels)
- 2*y_shift (y_shift in units of reslice file cubified voxels)
- 2*z_shift (z_shift in units of reslice file cubified voxels)
- x_scale
- y_scale
- z_scale

The reslice file voxel units referred to above are cubic (i.e., already interpolated to correct for reslice file voxel size anisotropy).

F.28.3.4 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using a traditional 9 parameter model is:

- pitch=0
- roll=0
- yaw=0
- x_shift=0
- y_shift=0
- z_shift=0
- x_scale=1
- y_scale=1
- z_scale=1

This results in the exact centers of the two files being aligned to one another.

F.28.3.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define a traditional 9 parameter transformation. Without explicit equations such as those provided above, the terms x_scale,y_scale, z_scale, pitch, roll, yaw, x-shift, y-shift, and z-shift are ambiguous. Other packages for registering or displaying images may not apply the transformations in the same order as the AIR package, so simple substitution of nominal parameters called "pitch","roll", etc. from other packages may not produce the desired result. In moving from standard file coordinates to reslice file coordinates, the AIR package performs x-,y-,and z-scaling, then rotations around the z-axis (yaw), x-axis (pitch), and y-axis (roll) in that order followed by translations x-shift, y-shift, and z-shift. Rotations operate around the exact centers of the files (before and after interpolation) and the sign conventions for both rotations and translations are arbitrary and may differ from those used in your alternative package. Shifts are defined in a version of reslice file space that has been interpolated to cubic voxels and are expressed in units of cubified voxels.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.28.4 Affine Transformations (12 parameters)

- transformation matrix
  - affine parameter matrix
- representation in .air files
- initialization files
- default initialization

- integration with other registration and display packages
- Programs that incorporate the affine transformation model:
  - alignlinear

F.28.4.1 TRANSFORMATION MATRIX

The affine model requires that lines that are parallel before transformation remain parallel after transformation. In the AIR package, the affine model is parameterized in terms of twelve parameters defined below. These parameters do not involve explicit definition of rotations, etc.

The affine transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as a homogenous transformation matrices:

(reslice file internal coordinates)=A*(standard file internal coordinates)

where

- A specifies the parameters of the affine transformation

F.28.4.1.1 Affine parameter matrix

$A =$

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & m \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

a,b,c,d,e,f,g,h,i,j,k and m are independent parameters

F.28.4.2 REPRESENTATION IN .AIR FILES

Since the .air file format requires that the matrix stored there represents a transformation that will convert coordinates from a version of the standard file that has been interpolated to cubic voxels, the matrix A must be modified before storage in a .air file.

This is accomplished by:

- dividing a, e and i by sxoom=(standard file voxel x size) / (smallest standard file voxel size)
- dividing b, f and j by syoom=(standard file voxel y size) / (smallest standard file voxel size)
- dividing c, g and k by szoom=(standard file voxel z size) / (smallest standard file voxel size)

F.28.4.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the affine model consist of the following in ASCII format:

- a
- b

- c
- d
- e
- f
- g
- h
- i
- j
- k
- m

F.28.4.4 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using an affine model is:

- a =sx_size / rx_size
- b=0
- c =0
- d =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- e=0
- f =sy_size / ry_size
- g =0
- h =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- i=0
- j =0
- k =sz_size / rz_size
- m =(rz_dim - sz_dim*(sz_size / rz_size)) / 2

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- sz_size is the voxel z size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- rz_size is the voxel z size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file

- sz_dim is the z dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file
- rz_dim is the z dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.28.4.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define an affine transformation. Without explicit equations such as those provided above, the terms a,b,c,d,e,f,g,h,i,j,k and m are ambiguous. Aside from d, h and m which specify x-, y- and z-axis shifts respectively in a version of the reslice file that has been interpolated to cubic voxels and which are expressed in units of cubified voxels, the physical meanings of the other parameters are difficult to interpret in isolation.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.28.5 Perspective Transformations (15 parameters, not supported for all cost functions)

- transformation matrix
  - perspective parameter matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages
- Programs that incorporate the perspective transformation model:
  - alignlinear

F.28.5.1 TRANSFORMATION MATRIX

The perspective model requires only that points that line on a line before transformation remain on a line after transformation. In the AIR package, the perspective model is parameterized in terms of fifteen parameters defined below. These parameters do not involve explicit definition of rotations, etc.

The perspective transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as a homogenous transformation matrix:

(reslice file internal coordinates)=Q*(standard file internal coordinates)

where

Qspecifies the parameters of the perspective transformation .

F.28.5.1.1 Perspective parameter matrix

$$Q = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & m \\ n & p & q & 1 \end{bmatrix}$$

where:

a,b,c,d,e,f,g,h,i,j,k,m,n,p and q are independent parameters

F.28.5.2 REPRESENTATION IN .AIR FILES

Since the .air file format requires that the matrix stored there represents a transformation that will convert coordinates from a version of the standard file that has been interpolated to cubic voxels, the matrix Q must be modified before storage in a .air file.

This is accomplished by:

- dividing a, e,i and n by sxoom=(standard file voxel x size) / (smallest standard file voxel size)
- dividing b, f, j and p by syoom=(standard file voxel y size) / (smallest standard file voxel size)
- dividing c, g, k and q by szoom=(standard file voxle z size) / (smallest standard file voxel size)

F.28.5.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the affine model consist of the following in ASCII format:

- a
- b
- c
- d
- e
- f
- g
- h
- i
- j
- k
- m
- n
- p
- q

F.28.5.4 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using an affine model is:

- a =sx_size / rx_size
- b=0
- c =0
- d =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- e=0
- f =sy_size / ry_size
- g =0
- h =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- i=0
- j =0
- k =sz_size / rz_size
- m =(rz_dim - sz_dim*(sz_size / rz_size)) / 2
- n =0
- p =0
- q =0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- sz_size is the voxel z size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- rz_size is the voxel z size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- sz_dim is the z dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file
- rz_dim is the z dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.28.5.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define a perspective transformation. Without explicit equations such as those provided above, the terms

a,b,c,d,e,f,g,h,i,j,k,m,n, p and q are ambiguous. Aside from d, h and m which specify x-, y- and z-axis shifts respectively in a version of the reslice file that has been interpolated to cubic voxels and which are expressed in units of cubified voxels, the physical meanings of the other parameters are difficult to interpret in isolation.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.29 3D NONLINEAR MODELS

F.29.1 3D First Order Nonlinear Transformation (12 Parameters)

A first order nonlinear transformation is actually the same as an affine linear transformation, but the model is implemented differently to make the parameters consistent with the parameters used by the nonlinear transformations.

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.29.1.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$x'=k_{01}+k_{02}x+k_{03}y+k_{04}z$

$x'=k_{05}+k_{06}x+k_{07}y+k_{08}z$

$x'=k_{09}+k_{10}x+k_{11}y+k_{12}z$

where:

k_{01}-k_{12} are independent parameters.

F.29.1.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the 3D first order nonlinear model consist of k_{01}-k_{12} in an ascii text file.

F.29.1.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for the 3D first order nonlinear model is:

- k_{01} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{02} =sx_size / rx_size
- k_{05} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{07} =sy_size / ry_size

- k_{09} =(rz_dim - sz_dim*(sz_size / rz_size)) / 2
- k_{12} =sz_size / rz_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- sz_size is the voxel z size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- rz_size is the voxel z size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- sz_dim is the z dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file
- rz_dim is the z dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.29.2 3D Second Order Nonlinear Transformation (30 parameters)

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.29.2.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$x'=k_{01}+k_{02}x+k_{03}y+k_{04}z+k_{05}x^2+k_{06}xy+k_{07}y^2+k_{08}xz+k_{09}yz+k_{10}z^2$

$x'=k_{11}+k_{12}x+k_{13}y+k_{14}z+k_{15}x^2+k_{16}xy+k_{17}y^2+k_{18}xz+k_{19}yz+k_{20}z^2$

$x'=k_{21}+k_{22}x+k_{23}y+k_{24}z+k_{25}x^2+k_{26}xy+k_{27}y^2+k_{28}xz+k_{29}yz+k_{30}z^2$

where:

k_{01}-k_{30} are independent parameters.

F.29.2.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files for consist of k_{01}-k_{30} in an ascii text file.

F.29.2.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- k_{01} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{02} =sx_size / rx_size
- k_{11} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{13} =sy_size / ry_size
- k_{21} =(rz_dim - sz_dim*(sz_size / rz_size)) / 2
- k_{24} =sz_size / rz_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- sz_size is the voxel z size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- rz_size is the voxel z size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- sz_dim is the z dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file

rz_dim is the z dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.29.3 3D Third Order Nonlinear Transformation (60 parameters)

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.29.3.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$$x'=k_{01}+k_{02}x+k_{03}y+k_{04}z+k_{05}x^2+k_{06}xy+k_{07}y^2+k_{08}xz+k_{09}yz+k_{10}z^2+k_{11}x^3+k_{12}x^2y+k_{13}xy^2+k_{14}y^3+k_{15}x^2z+k_{16}xyz+k_{17}y^2z+k_{18}xz^2+k_{19}yz^2+k_{20}z^3$$

$y'=k_{21}+k_{22}x+k_{23}y+k_{24}z+k_{25}x^2+k_{26}xy+k_{27}y^2+k_{28}xz+k_{29}yz+k_{30}z^2+k_{31}x^3+k_{32}x^2y+k_{33}xy^2+k_{34}y^3+k_{35}x^2z+k_{36}xyz+k_{37}y^2z+k_{38}xz^2+k_{39}yz^2+k_{40}z^3$

$z'=k_{41}+k_{42}x+k_{43}y+k_{44}z+k_{45}x^2+k_{46}xy+k_{47}y^2+k_{48}xz+k_{49}yz+k_{50}z^2+k_{51}x^3+k_{52}x^2y+k_{53}xy^2+k_{54}y^3+k_{55}x^2z+k_{56}xyz+k_{57}y^2z+k_{58}xz^2+k_{59}yz^2+k_{60}z^3$

where:

k_{01}-k_{60} are independent parameters.

F.29.3.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files consist of k_{01}-k_{60} in an ascii text file.

F.29.3.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- *k01 =(rx_dim - sx_dim*(sx_size / rx_size)) / 2*
- *k02 =sx_size / rx_size*
- *k21 =(ry_dim - sy_dim*(sy_size / ry_size)) / 2*
- *k23 =sy_size / ry_size*
- *k41 =(rz_dim - sz_dim*(sz_size / rz_size)) / 2*
- *k44 =sz_size / rz_size*
- *All other parameters=0*

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- sz_size is the voxel z size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- rz_size is the voxel z size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- sz_dim is the z dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file
- rz_dim is the z dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.29.4 3D Fourth Order Nonlinear Transformation (105 Parameters)

- transformation
- initialization files

- default initialization
- Programs that incorporate this model:
  - align_warp

F.29.4.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$$x'=k_{001}+k_{002}x+k_{003}y+k_{004}z+k_{005}x^2+k_{006}xy+k_{007}y^2+k_{008}xz+k_{009}yz+k_{010}z^2+k_{011}x^3+k_{012}x^2y+k_{013}xy^2+k_{014}y^3+k_{015}x^2z+k_{016}xyz+k_{017}y^2z+k_{018}xz^2+k_{019}yz^2+k_{020}z^3+k_{021}x^4+k_{022}x^3y+k_{023}x^2y^2+k_{024}xy^3+k_{025}y^4+k_{026}x^3z+k_{027}x^2yz+k_{028}xy^2z+k_{029}y^3z+k_{030}x^2z^2+k_{031}xyz^2+k_{032}y^2z^2+k_{033}xz^3+k_{034}yz^3+k_{035}z^4$$

$$y'=k_{036}+k_{037}x+k_{038}y+k_{039}z+k_{040}x^2+k_{041}xy+k_{042}y^2+k_{043}xz+k_{044}yz+k_{045}z^2+k_{046}x^3+k_{047}x^2y+k_{048}xy^2+k_{049}y^3+k_{050}x^2z+k_{051}xyz+k_{052}y^2z+k_{053}xz^2+k_{054}yz^2+k_{055}z^3+k_{056}x^4+k_{057}x^3y+k_{058}x^2y^2+k_{059}xy^3+k_{060}y^4+k_{061}x^3z+k_{062}x^2yz+k_{063}xy^2z+k_{064}y^3z+k_{065}x^2z^2+k_{066}xyz^2+k_{067}y^2z^2+k_{068}xz^3+k_{069}yz^3+k_{070}z^4$$

$$z'=k_{071}+k_{072}x+k_{073}y+k_{074}z+k_{075}x^2+k_{076}xy+k_{077}y^2+k_{078}xz+k_{079}yz+k_{080}z^2+k_{081}x^3+k_{082}x^2y+k_{083}xy^2+k_{084}y^3+k_{085}x^2z+k_{086}xyz+k_{087}y^2z+k_{088}xz^2+k_{089}yz^2+k_{090}z^3+k_{091}x^4+k_{092}x^3y+k_{093}x^2y^2+k_{094}xy^3+k_{095}y^4+k_{096}x^3z+k_{097}x^2yz+k_{098}xy^2z+k_{099}y^3z+k_{100}x^2z^2+k_{101}xyz^2+k_{102}y^2z^2+k_{103}xz^3+k_{104}yz^3+k_{105}z^4$$

where:

k_{001}-k_{105} are independent parameters.

F.29.4.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files consist of k_{001}-k_{105} in an ascii text file.

F.29.4.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- k_{001} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{002} =sx_size / rx_size
- k_{036} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{038} =sy_size / ry_size
- k_{071} =(rz_dim - sz_dim*(sz_size / rz_size)) / 2
- k_{074} =sz_size / rz_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- sz_size is the voxel z size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file

- rz_size is the voxel z size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- sz_dim is the z dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file
- rz_dim is the z dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.29.5 3D Fifth Order Nonlinear Transformation (168 Parameters)

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.29.5.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$$x'=k_{001}+k_{002}x+k_{003}y+k_{004}z+k_{005}x^2+k_{006}xy+k_{007}y^2+k_{008}xz+k_{009}yz+k_{010}z^2+k_{011}x^3+k_{012}x^2y+k_{013}xy^2+k_{014}y^3+k_{015}x^2z+k_{016}xyz+k_{017}y^2z+k_{018}xz^2+k_{019}yz^2+k_{020}z^3+k_{021}x^4+k_{022}x^3y+k_{023}x^2y^2+k_{024}xy^3+k_{025}y^4+k_{026}x^3z+k_{027}x^2yz+k_{028}xy^2z+k_{029}y^3z+k_{030}x^2z^2+k_{031}xyz^2+k_{032}y^2z^2+k_{033}xz^3+k_{034}yz^3+k_{035}z^4+k_{036}x^5+k_{037}x^4y+k_{038}x^3y^2+k_{039}x^2y^3+k_{040}xy^4+k_{041}y^5+k_{042}x^4z+k_{043}x^3yz+k_{044}x^2y^2z+k_{045}xy^3z+k_{046}y^4z+k_{047}x^3z^2+k_{048}x^2yz^2+k_{049}xy^2z^2+k_{050}y^3z^2+k_{051}x^2z^3+k_{052}xyz^3+k_{053}y^2z^3+k_{054}xz^4+k_{055}yz^4+k_{056}z^5$$

$$y'=k_{057}+k_{058}x+k_{059}y+k_{060}z+k_{061}x^2+k_{062}xy+k_{063}y^2+k_{064}xz+k_{065}yz+k_{066}z^2+k_{067}x^3+k_{068}x^2y+k_{069}xy^2+k_{070}y^3+k_{071}x^2z+k_{072}xyz+k_{073}y^2z+k_{074}xz^2+k_{075}yz^2+k_{076}z^3+k_{077}x^4+k_{078}x^3y+k_{079}x^2y^2+k_{080}xy^3+k_{081}y^4+k_{082}x^3z+k_{083}x^2yz+k_{084}xy^2z+k_{085}y^3z+k_{086}x^2z^2+k_{087}xyz^2+k_{088}y^2z^2+k_{089}xz^3+k_{090}yz^3+k_{091}z^4+k_{092}x^5+k_{093}x^4y+k_{094}x^3y^2+k_{095}x^2y^3+k_{096}xy^4+k_{097}y^5+k_{098}x^4z+k_{099}x^3yz+k_{100}x^2y^2z+k_{101}xy^3z+k_{102}y^4z+k_{103}x^3z^2+k_{104}x^2yz^2+k_{105}xy^2z^2+k_{106}y^3z^2+k_{107}x^2z^3+k_{108}xyz^3+k_{109}y^2z^3+k_{110}xz^4+k_{111}yz^4+k_{112}z^5$$

$$z'=k_{113}+k_{114}x+k_{115}y+k_{116}z+k_{117}x^2+k_{118}xy+k_{119}y^2+k_{120}xz+k_{121}yz+k_{122}z^2+k_{123}x^3+k_{124}x^2y+k_{125}xy^2+k_{126}y^3+k_{127}x^2z+k_{128}xyz+k_{129}y^2z+k_{130}xz^2+k_{131}yz^2+k_{132}z^3+k_{133}x^4+k_{134}x^3y+k_{135}x^2y^2+k_{136}xy^3+k_{137}y^4+k_{138}x^3z+k_{139}x^2yz+k_{140}xy^2z+k_{141}y^3z+k_{142}x^2z^2+k_{143}xyz^2+k_{144}y^2z^2+k_{145}xz^3+k_{146}yz^3+k_{147}z^4+k_{148}x^5+k_{149}x^4y+k_{150}x^3y^2+k_{151}x^2y^3+k_{152}xy^4+k_{153}y^5+k_{154}x^4z+k_{155}x^3yz+k_{156}x^2y^2z+k_{157}xy^3z+k_{158}y^4z+k_{159}x^3z^2+k_{160}x^2yz^2+k_{161}xy^2z^2+k_{162}y^3z^2+k_{163}x^2z^3+k_{164}xyz^3+k_{165}y^2z^3+k_{166}xz^4+k_{167}yz^4+k_{168}z^5$$

where:

k_{001}-k_{168} are independent parameters.

F.29.5.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files consist of k_{001}-k_{168} in an ascii text file.

F.29.5.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- k_{001} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{002} =sx_size / rx_size
- k_{057} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{059} =sy_size / ry_size
- k_{113} =(rz_dim - sz_dim*(sz_size / rz_size)) / 2
- k_{116} =sz_size / rz_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- sz_size is the voxel z size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- rz_size is the voxel z size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- sz_dim is the z dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file
- rz_dim is the z dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.30 2D LINEAR MODELS

F.30.1 2D Rigid Body Transformations (3 Parameters)

- transformation matrix
  - standard file interpolation matrix
  - standard file centering matrix
  - pixel size correction matrix
  - rigid body rotation matrix
  - rigid body translation matrix

  - reslice file inverse centering matrix
  - reslice file inverse interpolation matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages
- Programs that incorporate the rigid body transformation model:
  - alignlinear
  - manualreslice
  - alignpettopet (AIR 1.0)
  - alignmritopet (AIR 1.0)
  - alignpettomri (AIR 1.0)

F.30.1.1 TRANSFORMATION MATRIX

The 2D rigid body model requires that the real world Euclidean distance between any two coordinate locations to remain unchanged by the transformation. Since the AIR package allows anisotropic voxels sizes within a given file as well as different voxel sizes between files, these factors must be taken into account when applying a 2D rigid body transformation. In the AIR package, the 2D rigid body model is parameterized in terms of a rotation around the z-axis and translations along the x- and y- coordinate axes. In order to make these parameters more inituitive, the rotations of the rigid body transformation are defined as taking place around the centers of the files rather than the origin of the internal coordinate system (located at one corner of the file).

The 2D rigid body transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as the product of a series of homogenous transformation matrices:

(reslice file internal coordinates)=Zr*Cr*T*R*P*Cs*Zs*(standard file internal coordinates)

where

- Zs corrects for voxel size anisotropy in the standard file and is omitted when the reslice file is to be resampled to generate cubic voxels.
- Cs shifts the coordinate system to the center of the standard file
- P corrects for differences in pixel size in the two files
- R performs a rigid body rotation
- T performs a rigid body translation
- Cr shifts the coordinate system from the center back to one corner of the reslice file
- Zr corrects for voxel size anisotropy in the reslice file

F.30.1.1.1 Standard file interpolation matrix

Zs=

$$\begin{bmatrix} sxoom & 0 & 0 & 0 \\ 0 & syoom & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

sxoom=(standard file voxel x size) / (smallest standard file voxel size)
syoom=(standard file voxel y size) / (smallest standard file voxel size)

smallest standard file voxel size=min(standard file voxel x, y **and z** sizes)

This homogenous transformation matrix remaps coordinate locations in the standard file to new coordinates with cubic voxels. The origin remains at (0,0,0).

F.30.1.1.2 Standard file centering matrix

Cs=

$$\begin{bmatrix} 1 & 0 & 0 & -(sx\_dim-1)*sxoom/2 \\ 0 & 1 & 0 & -(sy\_dim-1)*syoom/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ where:

sx_dim=standard file x-dimension
sy_dim=standard file y-dimension
sxoom=(standard file voxel x size) / (smallest standard file voxel size)
syoom=(standard file voxel y size) / (smallest standard file voxel size)

smallest standard file voxel size=min(standard file voxel x, y **and z** sizes)

This homogenous coordinate transformation matrix shifts the origin from (0,0,0) to the exact center of the standard file.

F.30.1.1.3 Pixel size correction matrix

P=

$$\begin{bmatrix} ssize/rsize & 0 & 0 & 0 \\ 0 & ssize/rsize & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

ssize=min(standard file voxel x, y **and z** sizes)
rsize=min(reslice file voxel x, y **and z** sizes)

Coordinate units are modified by this homogenous coordinate transformation matrix to be equivalent to those used in an interpolated version of the reslice file.

F.30.1.1.4 Rigid body rotation matrix

R=

$$\begin{bmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

φ = rotation around the z - axis (yaw)

This homogenous coordinate transformation matrix performs rotations while preserving Euclidean distances between coordinate locations.

F.30.1.1.5 Rigid body translation matrix

T=

$$\begin{bmatrix} 1 & 0 & 0 & x\text{-shift} \\ 0 & 1 & 0 & y\text{-shift} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

x - shift = translation along the x - axis

y - shift = translation along the y - axis

This homogenous coordinate transformation matrix performs translations while preserving Euclidean distances between coordinate locations.

Note that the shifts have units of interpolated reslice file voxels

F.30.1.1.6 Reslice file inverse centering matrix

Cr=

$$\begin{bmatrix} 1 & 0 & 0 & (rx\_dim-1)*rxoom/2 \\ 0 & 1 & 0 & (ry\_dim-1)*ryoom/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

rx_dim=reslice file x-dimension
ry_dim=reslice file y-dimension
rxoom=(reslice file voxel x size) / (smallest reslice file voxel size)
ryoom=(reslice file voxel y size) / (smallest reslice file voxel size)

smallest reslice file voxel size=min(reslice file voxel x, y **and z** sizes)

The origin of the coordinate system is shifted from the center of the file to internal coordinate (0,0,0) of the reslice file by this homogenous coordinate transformation matrix.

F.30.1.1.7 Reslice file inverse interpolation matrix

Zr=

$$Zr=$$

$$\begin{bmatrix} 1/rxoom & 0 & 0 & 0 \\ 0 & 1/ryoom & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

rxoom=(reslice file voxel x size) / (smallest reslice file voxel size)
ryoom=(reslice file voxel y size) / (smallest reslice file voxel size)

smallest reslice file voxel size=min(reslice file voxel x, y **and z** sizes)

Cubic voxel coordinate locations are remapped to the actual voxel locations in the reslice file by this homogenous coordinate transformation matrix. The origin (0,0,0) remains unchanged.

F.30.1.2 REPRESENTATION IN .AIR FILES

Since the standard file interpolation matrix Zs is already implicit in the definition of the.air file transformation matrix, it is omitted from the matrix defined above when creating the .air file matrix: Zr*Cr*T*R*P*Cs.

F.30.1.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the rigid body model consist of the following in ASCII format:

- yaw (in radians)
- 2*x_shift (x_shift in units of reslice file voxels)
- 2*y_shift (y_shift in units of reslice file voxels)

The reslice file voxel units referred to above are cubic (i.e., already interpolated to correct for reslice file voxel size anisotropy).

F.30.1.4 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using a rigid body model is:

- yaw=0
- x_shift=0
- y_shift=0

This results in the exact centers of the two files being aligned to one another.

F.30.1.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define a rigid body transformation. Without explicit equations such as those provided above, the terms yaw, x-shift and y-shift are ambiguous. Other packages for registering or displaying images may not apply the transformations in the same order as the AIR package, so simple

substitution of nominal parameters called "yaw", etc. from other packages may not produce the desired result. In moving from standard file coordinates to reslice file coordinates, the AIR package performs rotations around the z-axis (yaw) followed by the translations x-shift and y-shift. Rotations operate around the exact centers of the files (before and after interpolation) and the sign conventions for both rotations and translations are arbitrary and may differ from those used in your alternative package. Shifts are defined in a version of reslice file space that has been interpolated to cubic voxels and are expressed in units of cubified voxels.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.30.2 2D Global Rescaling Transformations (4 Parameters)

- transformation matrix
  - standard file interpolation matrix
  - standard file centering matrix
  - pixel size correction matrix
  - rescaling matrix
  - rigid body rotation matrix
  - rigid body translation matrix
  - reslice file inverse centering matrix
  - reslice file inverse interpolation matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages
- Programs that incorporate the global rescaling transformation model:
  - alignlinear

F.30.2.1 TRANSFORMATION MATRIX

The 2D global rescaling model requires that the real world Euclidean distance between any two coordinate locations to by multiplied by a constant as a result of the transformation. Since the AIR package allows anisotropic voxels sizes within a given file as well as different voxel sizes between files, these factors must be taken into account when applying a 2D global rescaling transformation. In the AIR package, the global rescaling model is parameterized in terms of a rotation around the z-axis, translations along the x and y axes and a global rescaling term. In order to make these parameters more inituitive, the rotations of the global rescaling transformation are defined as taking place around the centers

of the files rather than the origin of the internal coordinate system (located at one corner of the file).

The global rescaling transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as the product of a series of homogenous transformation matrices:

(reslice file internal coordinates)=Zr*Cr*T*R*G*P*Cs*Zs*(standard file internal coordinates)

where

- Zs corrects for voxel size anisotropy in the standard file and is omitted when the reslice file is to be resampled to generate cubic voxels.
- Cs shifts the coordinate system to the center of the standard file
- P corrects for differences in pixel size in the two files
- G performs global rescaling
- R performs a rigid body rotation
- T performs a rigid body translation
- Cr shifts the coordinate system from the center back to one corner of the reslice file

F.30.2.1.1 Standard file interpolation matrix

See F.30.1.1.1 Standard file interpolation matrix above

F.30.2.1.2 Standard file centering matrix

See F.30.1.1.2 Standard file centering matrix above

F.30.2.1.3 Pixel size correction matrix

See F.30.1.1.3 Pixel size correction matrix above

F.30.2.1.4 Global rescaling matrix

G=

$$\begin{bmatrix} scale & 0 & 0 & 0 \\ 0 & scale & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This homogenous coordinate transformation matrix performs global rescaling.

F.30.2.1.5 Rigid body rotation matrix

See F.30.1.1.4 Rigid body rotation matrix above

F.30.2.1.6 Rigid body translation matrix

See F.30.1.1.5 Rigid body translation matrix above

F.30.2.1.7 Reslice file inverse centering matrix

See F.30.1.1.6 Reslice file inverse centering matrix above

F.30.2.1.8 Reslice file inverse interpolation matrix

See F.30.1.1.7 Reslice file inverse interpolation matrix above

F.30.2.2 REPRESENTATION IN .AIR FILES

Since the standard file interpolation matrix Zs is already implicit in the definition of the.air file transformation matrix, it is omitted from the matrix defined above when creating the .air file matrix: Zr*Cr*T*R*G*P*Cs.

F.30.2.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the global rescaling model consist of the following in ASCII format:

- yaw (in radians)
- 2*x_shift (x_shift in units of reslice file cubified voxels)
- 2*y_shift (y_shift in units of reslice file cubified voxels)
- scale

The reslice file voxel units referred to above are cubic (i.e., already interpolated to correct for reslice file voxel size anisotropy).

F.30.2.4 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using a global rescaling model is:

- yaw=0
- x_shift=0
- y_shift=0
- scale=1

This results in the exact centers of the two files being aligned to one another.

F.30.2.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define a 2D global rescaling transformation. Without explicit equations such as those provided above, the terms scale, yaw, x-shift and y-shift are ambiguous. Other packages for registering or displaying images may not apply the transformations in the same order as the AIR package, so simple substitution of nominal parameters called "yaw", etc. from other packages may not produce the desired result. In moving from standard file coordinates to reslice file coordinates, the AIR package performs scaling and rotation around the z-axis (yaw) followed by the translations x-shift and y-shift. Rotations operate around the exact centers of the files (before and after interpolation) and the sign conventions for both rotations and translations are arbitrary and may differ from those used in your alternative package. Shifts are defined in a version of reslice file space that has been interpolated to cubic voxels and are expressed in units of cubified voxels.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.30.3 2D Fixed Determinant Transformations (5 Parameters)

- transformation matrix
  - standard file interpolation matrix
  - pixel size correction matrix
  - fixed determinant matrix
  - reslice file inverse interpolation matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages
- Programs that incorporate the 2D fixed determinant transformation model:
  - alignlinear

F.30.3.1 TRANSFORMATION MATRIX

The 2D fixed determinant model requires that the real world transformation have a determinant of one (i.e., areas are unchanged by transformation).Since the AIR package allows anisotropic voxels sizes within a given file as well as different voxel sizes between files, these factors must be taken into account when applying a 2D fixed determinant transformation. In the AIR package, the 2D fixed determinant model is parameterized in terms of five parameters.

The 2D fixed determinant transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as the product of a series of homogenous transformation matrices:

(reslice file internal coordinates)=Zr*D*P*Zs*(standard file internal coordinates)

where

- Zs corrects for voxel size anisotropy in the standard file and is omitted when the reslice file is to be resampled to generate cubic voxels.
- P corrects for differences in pixel size in the two files
- D is a matrix with a determinant of one
- Zr corrects for voxel size anisotropy in the reslice file

F.30.3.1.1 Standard file interpolation matrix

See F.30.1.1.1 Standard file interpolation matrix above

F.30.3.1.2 Pixel size correction matrix

See F.30.1.1.3 Pixel size correction matrix above

F.30.3.1.3 Fixed determinant matrix

D=

$$\begin{bmatrix} a & b & 0 & c \\ d & E & 0 & f \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

a, b, c, d, and f are related to the formal parameters as specified below
E=(1+b*d) / a

Since a*E - b*d=1, this homogenous coordinate transformation matrix performs affine distortions subject to the restriction that areas are not changed.

F.30.3.1.4 Reslice file inverse interpolation matrix

See F.30.1.1.7 Reslice file inverse interpolation matrix above

F.30.3.2 REPRESENTATION IN .AIR FILES

Since the standard file interpolation matrix Zs is already implicit in the definition of the.air file transformation matrix, it is omitted from the matrix defined above when creating the .air file matrix: Zr*D*P.

F.30.3.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the 2D fixed determinant model consist of the following formal parameters in ASCII format:

- a*(standard file voxel x size) / (reslice file voxel x size)
- b*(standard file voxel y size) / (reslice file voxel x size)
- c
- d*(standard file voxel x size) / (reslice file voxel y size)
- f

F.30.3.4 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using a 2D fixed determinant model is:

- a=1
- b=0
- c=(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- d=0
- f=(ry_dim - sy_dim*(sy_size / ry_size)) / 2

This results in the exact centers of the two files being aligned to one another.

F.30.3.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define a 2D fixed determinant transformation. Without explicit equations such as those provided above, the terms a,b,c,d and f

are ambiguous. Aside from c and f which specify x- and y-axis shifts respectively in a version of the reslice file that has been interpolated to cubic voxels and which are expressed in units of cubified voxels, the physical meanings of the other parameters are difficult to interpret in isolation.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.30.4 2D Affine Transformations (6 Parameters)

- transformation matrix
  - 2D affine parameter matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages
- Programs that incorporate the affine transformation model:
  - alignlinear

F.30.4.1 TRANSFORMATION MATRIX

The 2D affine model requires that lines that are parallel before transformation remain parallel after transformation. In the AIR package, the 2D affine model is parameterized in terms of six parameters defined below. These parameters do not involve explicit definition of rotations, etc.

The 2D affine transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as a homogenous transformation matrices:

(reslice file internal coordinates)=A*(standard file internal coordinates)

where

- A specifies the parameters of the affine transformation

F.30.4.1.1 Affine parameter matrix

$A=$

$$\begin{bmatrix} a & b & 0 & c \\ d & e & 0 & f \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

a,b,c,d,e and f are independent parameters

F.30.4.2 REPRESENTATION IN .AIR FILES

Since the .air file format requires that the matrix stored there represents a transformation that will convert coordinates from a version of the standard file that has been interpolated to cubic voxels, the matrix A must be modified before storage in a .air file.

This is accomplished by:

- dividing a and d by sxoom=(standard file voxel x size) / (smallest standard file voxel size)
- dividing b and e by syoom=(standard file voxel y size) / (smallest standard file voxel size)

Note: The voxel z size is included when determining the smallest standard file voxel size

---

F.30.4.3 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the affine model consist of the following in ASCII format:

- a
- b
- c
- d
- e
- f

F.30.4.4 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using an affine model is:

- a =sx_size / rx_size
- b=0
- c =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- d =0
- e=sy_size / ry_size
- f =(ry_dim - sy_dim*(sy_size / ry_size)) / 2

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file

- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.30.4.5 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define an affine transformation. Without explicit equations such as those provided above, the terms a,b,c,d,e and f are ambiguous. Aside from c and f which specify x- and y-axis shifts respectively in a version of the reslice file that has been interpolated to cubic voxels and which are expressed in units of cubified voxels, the physical meanings of the other parameters are difficult to interpret in isolation.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.30.5 2D Perspective Transformations (8 parameters, not supported for all cost functions)

- transformation matrix
  - 2D perspective parameter matrix
- representation in .air files
- initialization files
- default initialization
- integration with other registration and display packages
- Programs that incorporate the perspective transformation model:
  - alignlinear

F.30.5.1 TRANSFORMATION MATRIX

The perspective model requires only that points that line on a line before transformation remain on a line after transformation. In the AIR package, the perspective model is parameterized in terms of eight parameters defined below. These parameters do not involve explicit definition of rotations, etc.

The perspective transformation for converting from an internal coordinate in the standard file to the corresponding internal coordinate in the reslice file is best expressed as a homogenous transformation matrix:

(reslice file internal coordinates)=Q*(standard file internal coordinates)

where

- Q specifies the parameters of the perspective transformation

F.30.5.2 PERSPECTIVE PARAMETER MATRIX

$$Q = \begin{bmatrix} a & b & 0 & c \\ d & e & 0 & f \\ 0 & 0 & 1 & 0 \\ g & h & 0 & 1 \end{bmatrix}$$

where:

a,b,c,d,e,f,g and h are independent parameters

F.30.5.3 REPRESENTATION IN .AIR FILES

Since the .air file format requires that the matrix stored there represents a transformation that will convert coordinates from a version of the standard file that has been interpolated to cubic voxels, the matrix Q must be modified before storage in a .air file.

This is accomplished by:

- dividing a, d and g by sxoom=(standard file voxel x size) / (smallest standard file voxel size)
- dividing b, e and h by syoom=(standard file voxel y size) / (smallest standard file voxel size) Note that the standard file voxel z size is considered in determining the smallest standard file voxel size.

F.30.5.4 REPRESENTATION IN INITIALIZATION FILES

Initialization files for the 2D perspective model consist of the following in ASCII format:

- a
- b
- c
- d
- e
- f
- g
- h

F.30.5.5 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization for programs using an 2D perspective model is:

- a =sx_size / rx_size
- b=0
- c =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- d =0

- e=sy_size / ry_size
- f =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- g =0
- h =0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.30.5.6 OTHER REGISTRATION AND DISPLAY PACKAGES

There are many different ways to define a 2D perspective transformation. Without explicit equations such as those provided above, the terms a,b,c,d,e,f,g and h are ambiguous. Aside from c and f which specify x- and y-axis shifts respectively in a version of the reslice file that has been interpolated to cubic voxels and which are expressed in units of cubified voxels, the physical meanings of the other parameters are difficult to interpret in isolation.

If your alternative package generates a linear algebraic transformation matrix of its own, don't forget that transformation matrices are dependent upon the coordinate system used and that the AIR internal coordinate system used to define transformation matrices may differ from that of your alternative package.

F.31 2D NONLINEAR MODELS

F.31.1 2D First Order Nonlinear Transformation (6 Parameters)

A first order nonlinear transformation is actually the same as an affine linear transformation, but the model is implemented differently to make the parameters consistent with the parameters used by the nonlinear transformations.

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.31.1.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$x'=k_{01}+k_{02}x+k_{03}y$

$x'=k_{04}+k_{05}x+k_{06}y$

where:

k_{01}-k_{06} are independent parameters.

F.31.1.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files consist of k_{01}-k_{06} in an ascii text file.

F.31.1.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- k_{01} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{02} =sx_size / rx_size
- k_{04} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{06} =sy_size / ry_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.31.2 2D Second Order Nonlinear Transformation (12 parameters)

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.31.2.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$x'=k_{01}+k_{02}x+k_{03}y+k_{04}x^2+k_{05}xy+k_{06}y^2$

$y'= k_{07}+k_{08}x+k_{09}y+k_{10}x^2+k_{11}xy+k_{12}y^2$

where:

k_{01}-k_{12} are independent parameters.

F.31.2.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files for consist of k_{01}-k_{12} in an ascii text file.

F.31.2.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- k_{01} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{02} =sx_size / rx_size
- k_{07} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{09} =sy_size / ry_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.31.3 2D Third Order Nonlinear Transformation (20 parameters)

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.31.3.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$x'=k_{01}+k_{02}x+k_{03}y+k_{04}x^2+k_{05}xy+k_{06}y^2+k_{07}x^3+k_{08}x^2y+k_{09}xy^2+k_{10}y^3$

$y'=k_{11}+k_{12}x+k_{13}y+k_{14}x^2+k_{15}xy+k_{16}y^2+k_{17}x^3+k_{18}x^2y+k_{19}xy^2+k_{20}y^3$

where:

k_{01}-k_{20} are independent parameters.

F.31.4 Representation in initialization files

Initialization files for consist of k_{01}-k_{20} in an ascii text file.

F.31.4.1 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- k_{01} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{02} =sx_size / rx_size
- k_{11} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{13} =sy_size / ry_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.32 2D FOURTH ORDER NONLINEAR TRANSFORMATION (30 PARAMETERS)

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.32.1.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$$x'=k_{01}+k_{02}x+k_{03}y+k_{04}x^2+k_{05}xy+k_{06}y^2+k_{07}x^3+k_{08}x^2y+k_{09}xy^2+k_{10}y^3+k_{11}x^4+k_{12}x^3y+k_{13}x^2y^2+k_{14}xy^3+k_{15}y^4$$

$$y'=k_{16}+k_{17}x+k_{18}y+k_{19}x^2+k_{20}xy+k_{21}y^2+k_{22}x^3+k_{23}x^2y+k_{24}xy^2+k_{25}y^3+k_{26}x^4+k_{27}x^3y+k_{28}x^2y^2+k_{29}xy^3+k_{30}y^4$$

where:

k_{01}-k_{30} are independent parameters.

F.32.1.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files for consist of k_{01}-k_{30} in an ascii text file.

F.32.1.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- k_{01} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{02} =sx_size / rx_size
- k_{16} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{18} =sy_size / ry_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file
- This results in the exact centers of the two files being aligned to one another.

F.32.2 2D Fifth Order Nonlinear Transformation (42 parameters)

- transformation
- initialization files
- default initialization
- Programs that incorporate this model:
  - align_warp

F.32.2.1 TRANSFORMATION

Given a coordinate (x,y,z) in the standard file, the coordinates of the corresponding voxel in the reslice file (x',y',z') are given by the equations:

$$x'=k_{01}+k_{02}x+k_{03}y+k_{04}x^2+k_{05}xy+k_{06}y^2+k_{07}x^3+k_{08}x^2y+k_{09}xy^2+k_{10}y^3+k_{11}x^4+k_{12}x^3y+k_{13}x^2y^2+k_{14}xy^3+k_{15}y^4+k_{16}x^5+k_{17}x^4y+k_{18}x^3y^2+k_{19}x^2y^3+k_{20}xy^4+k_{21}y^5$$

$$y'=k_{22}+k_{23}x+k_{24}y+k_{25}x^2+k_{26}xy+k_{27}y^2+k_{28}x^3+k_{29}x^2y+k_{30}xy^2+k_{31}y^3+k_{32}x^4+k_{33}x^3y+k_{34}x^2y^2+k_{35}xy^3+k_{36}y^4+k_{37}x^5+k_{38}x^4y+k_{39}x^3y^2+k_{40}x^2y^3+k_{41}xy^4+k_{42}y^5$$

where:

k_{01}-k_{42} are independent parameters.

F.32.2.2 REPRESENTATION IN INITIALIZATION FILES

Initialization files for consist of k_{01}-k_{42} in an ascii text file.

F.32.2.3 DEFAULT INITIALIZATION

If no initialization file is specified, the default initialization is:

- k_{01} =(rx_dim - sx_dim*(sx_size / rx_size)) / 2
- k_{02} =sx_size / rx_size
- k_{22} =(ry_dim - sy_dim*(sy_size / ry_size)) / 2
- k_{24} =sy_size / ry_size
- All other parameters=0

where:

- sx_size is the voxel x size of the standard file
- sy_size is the voxel y size of the standard file
- rx_size is the voxel x size of the reslice file
- ry_size is the voxel y size of the reslice file
- sx_dim is the x dimension of the standard file
- sy_dim is the y dimension of the standard file
- rx_dim is the x dimension of the reslice file
- ry_dim is the y dimension of the reslice file

This results in the exact centers of the two files being aligned to one another.

F.33 EIGHT AND SIXTEEN BIT IMAGES IN THE AIR PACKAGE

- General comments
- Internal and external data representation
  - importing data
  - internal data
  - exporting data
- External data types
  - Type 0
  - Type 1
  - Type 2
  - Type 3
- Import mapping of voxel values
  - into 8 bit version of AIR package

- into 16 bit version of AIR package
- Export mapping of voxel values
  - from 8 bit version of AIR package
  - from 16 bit version of AIR package

F.33.1 General comments

If you plan to use the AIR package with 16 bit data, or if you have compiled the AIR package in 16 bit format, you are strongly encouraged to read this page carefully. The AIR package remaps voxel values in the process of importing and exporting data when 16 bit formatting is involved. Successful use of the AIR package with 16 bit data requires a detailed understanding of these issues. Even if you fully understand the C representation of signed and unsigned integers, the AIR package handling of these issues is sufficiently idiosyncratic that you will need to review the information in this document. At a minimum, you should review illustrations showing the import and export mapping of voxel values for 8 bit and 16 bit versions of the AIR package.

The AIR package was originally designed for use with 8 bit data. Indeed, 8 bit representation is generally sufficient for automated registration of images. The primary justification for utilizing 16 bit data is the need for higher precision for additional data analysis that will occur after image registration is completed. Consequently, there are two major options for dealing with 16 bit data:

1. Use a 16 bit version of the entire AIR package for all procedures.

2. Use an 8 bit version of AIR to derive .air files (which are independent of whether the package is compiled in 8 or 16 bit mode) and then use a 16 bit version of AIR to apply these .air files to your data. (This is the approach that I generally recommended).

Both of these approaches will generate registered 16 bit images. With the second approach, RAM requirements of the automated registration programs are cut in half and computation time is decreased.

The primary disadvantage is that care must be taken to assure that the 16 bit values will be converted into 8 bit values in a way that maximizes the dynamic range of the converted images. The AIR package provides a mechanism for mapping 16 bit values to 8 bit values at the time that the images are loaded, making it unnecessary to store separate 8 bit versions of files on disk. Sixteen bit data can be represented on disk in a number of different formats, and it is critical that the format used to store the images to be registered be correctly identified. The AIR package cannot determine on its own what convention has been used for storing data, and **the user is responsible for correctly specifying how data in external files that were not generated by the AIR package is represented.** It is especially important that pixels that should be displayed as black by your image display program (assuming that black represents the background, i.e., not part of the object of interest) be mapped to a value of zero internally in AIR. The information below should help you to assure that this is the case.

When a 16 bit version of AIR package is compiled, a decision must be made about how data in 16 bit files generated by the AIR package will be represented.

The package must be recompiled if a different output representation is desired. **The person compiling the AIR package is responsible for correctly specifying how data in 16 bit files generated by the AIR package is represented**.

F.33.2 Internal and external data representation

In order to understand how the AIR package deals with the number of bits/pixel, it is necessary to make a distinction between three different types of data representation:

- external data to be loaded (imported) into the AIR package
- internal representation of data *within* the AIR package
- external data written out (exported) by the AIR package

F.33.2.1 EXTERNAL DATA TO BE LOADED (IMPORTING DATA)

All versions of the AIR package (8 or 16 bit) are able to load external data stored in 8 bit and 16 bit formats. The programs use the header files to determine whether a given image file contains 8 bit or 16 bit data. The external values stored in the image file are mapped onto the AIR package's internal representation according to preset rules designed to optimize use of the internal representation's dynamic range. The import mapping of 16 bit values to 8 bit values can be optimized by specifying the largest value represented in the 16 bit images. Three different types of external 16 bit data representation are supported. The specific 16 bit data type is specified in the header file, and an incorrect designation of the data type in the header file will result in an inappropriate internal representation of the data.

F.33.2.2 INTERNAL REPRESENTATION OF DATA

Prior to compiling the AIR package, you must configure the AIR.h file to specify whether the internal data representation will be 8 bits/pixel or 16 bits/pixel. The internal representation cannot be changed without recompiling the package. If 8 bits/pixel are specified, internal values of 0-255 are used. For 16 bits/pixel, internal values of 0-65535 are used. In order for automated registration to be successful, it is critical that all "black" pixels (i.e., "zero"-valued and undefined pixels) be mapped to an internal value of zero. Depending upon the external data types, voxel values may be systematically remapped when importing data into and when exporting data out of the AIR package.

F.33.2.3 EXTERNAL DATA CREATED BY THE AIR PACKAGE (EXPORTING DATA)

If you compile the AIR package in 8 bit mode, your output images will always have 8 bits/pixel. If you compile the AIR package in 16 bit mode, your output images will always have 16 bits/pixel. When you compile in 16 bit mode, you must also specify in the AIR.h file how you want 16 bit output data to be represented. Internal data will be mapped onto external values based upon this specification. You must choose only one of the three possible representations (these are the same three choices available for imported 16 bit data) when you compile the AIR package, and you must recompile the package if you want to change the output data representation. The AIR package cannot "remember" the file format that was used to load an image in order to specify an output using the

same format. It can only generate the one file format that was specified in the AIR.h file when the package was compiled.

F.33.3 External data types

The AIR package defines four different external data types. When external data is loaded, the data type is determined based on the three values stored in the header files. The header "bits/pixel" field is used to distinguish 8 bit data from 16 bit data, and the header "global maximum" and "global minimum" fields are used to distinguish the three different types of 16 bit data. If the AIR package was compiled using an 8 bit internal representation, the header "global maximum" is also used to optimize the conversion of 16 bit to 8 bit values. When the AIR package creates a file, it uses the format specified when the package was compiled and creates header files that correctly identify the storage format. You can use the program scanheader to see what values are stored for "bits/pixel", "global maximum" and "global minimum" and thereby deduce what type of data is being specified by the header. When you create a new header using makeaheader, you must explicitly identify the data type.

The following descriptions of the four external data types may be easier to understand if you review the illustrations that show how the various data types map onto eight and sixteen bit internal representations.

F.33.3.1 TYPE 0 DATA

- 8 bits per pixel
- Values correspond to to C "unsigned characters"
- Has defined values in the range 0 to 255
- A value of 0 corresponds to a "black" pixel and will be mapped to 0 internally by the AIR package
- Header global maximum and global minimum are irrelevant
- Corresponds exactly to the AIR 8 bit internal represenation

F.33.3.2 TYPE 1 DATA

- 16 bits per pixel
- Values correspond to C "unsigned short ints"
- Has defined values in the range 0 to 65535
- A value of 0 corresponds to a "black" pixel and will be mapped to 0 internally by the AIR package
- Has a header global minimum greater than or equal to 0
- Has a header global maximum greater than 32767
- Corresponds exactly to the AIR 16 bit internal representation

F.33.3.3 TYPE 2 DATA

- 16 bits per pixel
- Values correspond to C "short ints"

- Has defined values in the range 0 to 32767
- All negative values are undefined, correspond to "black" pixels, and will be mapped to 0 internally by the AIR package
- Has a header global minimum greater than or equal to 0
- Has a header global maximum greater than 0 and less than or equal to 32767

F.33.3.4 TYPE 3 DATA

- 16 bits per pixel
- Values correspond to C "short ints"
- Has defined values in the range -32768 to 32767
- A value of -32768 corresponds to a "black" pixel and will be mapped to 0 internally by the AIR package
- Has a header global minimum less than 0
- Has a header global maximum greater than -32768

F.33.4 Import mapping of voxel values

Pixel Value Remapping for an 8 Bit Version of AIR

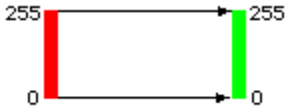
When the AIR package is compiled, it is compiled either as an 8 bit or as a 16 bit version. The information on this page describes the pixel value remapping behavior of AIR when it is compiled in 8 bit format. All 8 bit versions of AIR represent pixel values internally using numbers ranging from zero to 255. Your image generation and display software may require that data be represented differently, so the AIR package allows pixel values to be systematically remapped when loading data from disk. Pixel values are not modified by an 8 bit version of AIR when saving data to disk.

Pixel Value Remapping for a 16 Bit Version of AIR

When the AIR package is compiled, it is compiled either as an 8 bit or as a 16 bit version. The information on this page describes the pixel value remapping behavior of AIR when it is compiled in 16 bit format. All 16 bit versions of AIR represent pixel values internally using numbers ranging from zero to 65535. Your image generation and display software may require that data be represented differently, so the AIR package allows pixel values to be systematically remapped when loading data from disk or saving data to disk.

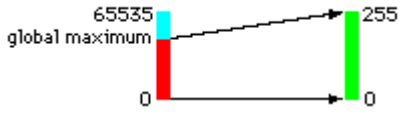
F.33.4.1 LOADING DATA FROM DISK INTO AN 8 BIT VERSION OF AIR

- When loading type 0 data (8 bit data) from disk, an 8 bit version of the AIR package will:

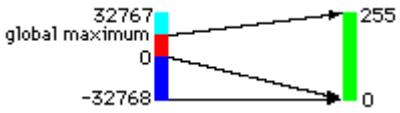


make no adjustment.

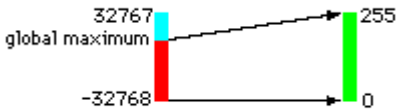
- When loading 16 bit type 1 data from disk, an 8 bit version of the AIR package will:



  - multiply by (65535 / (global max))
  - discard the fractional component
  - divide by 256
  - discard the fractional component
  - note that "global maximum" is stored in the image header and can be changed using the program setheadermax
- When loading 16 bit type 2 data from disk, an 8 bit version of the AIR package will:



  - set negative values to zero
  - multiply by 2
  - multiply by (32767/(global max))
  - discard the fractional component
  - divide by 256
  - discard the fractional component
  - note that "global maximum" is stored in the image header and can be changed using the program setheadermax
- When loading 16 bit type 3 data from disk, an 8 bit version of the AIR package will:



  - add 32768
  - multiply by (65535/(global max+32768))
  - discard the fractional component
  - divide by 256
  - discard the fractional component

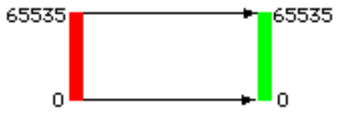
- note that "global maximum" is stored in the image header and can be changed using the program setheadermax

F.33.4.2 LOADING DATA FROM DISK INTO A 16 BIT VERSION OF AIR

- When loading type 0 data (8 bit data) from disk, a 16 bit version of the AIR package will:

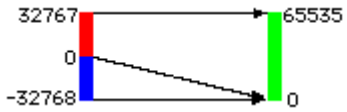
  - multiply by 256

- When loading16 bit type 1 data from disk, a 16 bit version of the AIR package will:



  - make no adjustment

- When loading 16 bit type 2 data from disk, a 16 bit version of the AIR package will:



  - set negative values to zero
  - multiply by 2

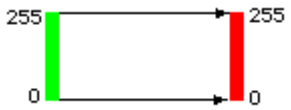
- When loading 16 bit type 3 data from disk, a 16 bit version of the AIR package will:

  - add 32768

F.33.5 Export mapping of voxel values

F.33.5.1 SAVING DATA FROM AN 8 BIT VERSION OF AIR

- All data (except for binary files) generated by an 8 bit version AIR package will be saved to disk as 8 bit values ranging from zero to 255. Consequently, the output data type (always 8 bit) will not necessarily match the input data type (which can be any 8 or 16 bit data type).



- ➢ no adjustment is made

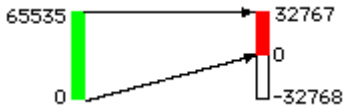
F.33.5.2 SAVING DATA FROM A 16 BIT VERSION OF AIR

When a 16 bit version of the AIR package is compiled, a default 16 bit storage data type is selected (the storage data type is specified in the AIR.h file). **All** data (except for binary files) generated by that compiled version of the AIR package will be saved to disk using the data type that was specified at compilation. Consequently, the output data type (which is fixed for any given compilation) will not necessarily match the input data type (which can be any data type).

- If a 16 bit version of AIR was compiled to generate external type 1 data

- ➢ no adjustment is made when writing data to disk

- If a 16 bit version of AIR was compiled to generate external type 2 data



- ➢ values are divided by 2 before writing them to disk

- If a 16 bit version of AIR was compiled to generate external type 3 data

- ➢ 32768 is subtracted from values before writing them to disk

F.34 GENERAL DISCUSSION OF HOMOGENOUS COORDINATES

- 4x1 homogenous coordinate vectors
- 4x4 homogenous coordinate matices
- translations
- rotations
- rescaling
- perspective

Homogenous coordinates utilize a mathematical trick to embed three-dimensional coordinates and transformations into a four-dimensional matrix format. As a result, inversions or combinations of linear transformations are simplified to inversion or multiplication of the corresponding matrices. Homogenous coordinates also make it possible to define perspective transformations.

F.34.1 4x1 Homogenous Coordinate Vectors

Instead of representing each point (x,y,z) in three-dimensional space with a single three-dimensional vector:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

homogenous coordinates allow each point (x,y,z) to be represented by any of an infinite number of four dimensional vectors:

$$\begin{bmatrix} T*x \\ T*y \\ T*z \\ T \end{bmatrix}$$

The three-dimensional vector corresponding to any four-dimensional vector can be computed by dividing the first three elements by the fourth, and a four-dimensional vector corresponding to any three-dimensional vector can be created by simply adding a fourth element and setting it equal to one.

Many textbooks define homogenous coordinates in such a way that points are represented by 1x4 vectors:

[T*x T*y T*z T]

instead of 4x1 vectors. This definition is not used in the AIR package and results in different 4x4 homogenous coordinate transformation matrices than those described below.

F.34.2 4x4 Homogenous Coordinate Transformation Matrices

Homogenous coordinate transformation matrices operate on four-dimensional homogenous coordinate vector representations of traditional three-dimensional coordinate locations. Any three-dimensional linear transformation (rotation, translation, skew, perspective distortion) can be represented by a 4x4 homogenous coordinate transformation matrix. In fact, because of the redundant representation of three space in a homogenous coordinate system, an infinite number of different 4x4 homogenous coordinate transformation matrices are available to perform any given linear transformation. This redundancy can be eliminated to provide a unique representation by dividing all elements of a 4x4 homogenous transformation matrix by the last element (which will become equal to one). This means that a 4x4 homogenous transformation matrix can incorporate as many as 15 independent parameters. The generic format representation of a homogenous transformation equation for mapping the three dimensional coordinate (x,y,z) to the three-dimensional coordinate (x',y',z') is:

$$\begin{bmatrix} T'\cdot x' \\ T'\cdot y' \\ T'\cdot z' \\ T' \end{bmatrix} = \begin{bmatrix} T''\cdot a & T''\cdot b & T''\cdot c & T''\cdot d \\ T''\cdot e & T''\cdot f & T''\cdot g & T''\cdot h \\ T''\cdot i & T''\cdot j & T''\cdot k & T''\cdot m \\ T''\cdot n & T''\cdot p & T''\cdot q & T'' \end{bmatrix} * \begin{bmatrix} T\cdot x \\ T\cdot y \\ T\cdot z \\ T \end{bmatrix}$$

If any two matrices or vectors of this equation are known, the third matrix (or vector) can be computed and then the redundant T element in the solution can be eliminated by dividing all elements of the matrix by the last element.

Various transformation models can be used to constrain the form of the matrix to tranformations with fewer degrees of freedom.

In many textbooks, you will find homogenous tranformation matrices defined such that 1x4 homogenous coordinate vectors are placed to the left of the 4x4 homogenous coordinate transformation matrix and multiplied. This format is not supported in the AIR package and the difference accounts for the fact that translations are represented in the fourth column and perspective distortions in the fourth row of AIR homologous transformation matrices rather than vice versa.

F.34.3 Translations

Translations can be represented by the 4x4 homogenous coordinate transformation matrix:

$$\begin{bmatrix} 1 & 0 & 0 & \text{x-shift} \\ 0 & 1 & 0 & \text{y-shift} \\ 0 & 0 & 1 & \text{z-shift} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

x-shift=translation along the x-axis
y-shift=translation along the y-axis
z-shift=translation along the z-axis

F.34.4 Rotations

A series of rotations (in the order [roll matrix]*[pitch matrix]*[yaw matrix]) can be represented by the 4x4 homogenous coordinate transformation matrix:

$$\begin{bmatrix} (\cos\varphi*\cos\phi+\sin\varphi*\sin\theta*\sin\phi) & (\sin\varphi*\cos\phi-\cos\varphi*\sin\theta*\sin\phi) & (\cos\theta*\sin\phi) & 0 \\ (-\sin\varphi*\cos\theta) & (\cos\varphi*\cos\theta) & (\sin\theta) & 0 \\ (\sin\varphi*\sin\theta*\cos\phi-\cos\varphi*\sin\phi) & (-\cos\varphi*\sin\theta*\cos\phi-\sin\varphi*\sin\phi) & (\cos\theta*\cos\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

θ=rotation around the x-axis (pitch)
ϕ=rotation around the y-axis (roll)
φ=rotation around the z-axis (yaw)

F.34.5 Rescaling

Rescaling along the major axes can be represented by the 4x4 homogenous coordinate transformation matrix:

$$\begin{bmatrix} xscale & 0 & 0 & 0 \\ 0 & yscale & 0 & 0 \\ 0 & 0 & zscale & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

xscale=rescaling along standard file x dimension
yscale=rescaling along standard file y dimension
zscale=rescaling along standard file z dimension

F.34.6 Perspective

Perspective distortion is achieved by applying the 4x4 homogenous coordinate transformation matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/xview & 1/yview & 1/zview & 1 \end{bmatrix}$$

where:

xview=x coordinate from which image is viewed
yview=y coordinate from which image is viewed
zview=z coordinate from which image is viewed

F.35 PRIMARY REFERENCES

One or more of the following references should be cited in any paper based on data registered using the AIR package or programs utilizing the AIR library.

- Woods RP, Cherry SR, Mazziotta JC. Rapid automated algorithm for aligning and reslicing PET images. Journal of Computer Assisted Tomography 1992;16:620-633.

The original AIR manuscript validating the intramodality method implemented in AIR 1.0.

- Woods RP, Mazziotta JC, Cherry SR. MRI-PET registration with automated algorithm. Journal of Computer Assisted Tomography 1993;17:536-546.

The method used for intermodality registration as implemented in AIR 1.0

- Woods RP, Grafton ST, Holmes CJ, Cherry SR, Mazziotta JC. Automated image registration: I. General methods and intrasubject, intramodality validation. Journal of Computer Assisted Tomography 1998;22:141-154.

Description of AIR 3.0 and validation for intrasubject, intramodality registration of MRI data and intrasubject, intramodality registration of PET data.

- Woods RP, Grafton ST, Watson JDG, Sicotte NL, Mazziotta JC. Automated image registration: II. Intersubject validation of linear and nonlinear models. Journal of Computer Assisted Tomography 1998;22:155-165.

Validation of AIR 3.0 for intersubject registration.

F.36 SECONDARY REFERENCES

The following references provide additional validation of AIR.

- Strother SC, Anderson JR, Xu XL, Liow JS, Bonar DC, Rottenberg, DA. Quantitative comparisons of image registration techniques based on high-resolution MRI of the brain. Journal of Computer Assisted Tomography 1994;18:954-62.

A comparative study using simulations that included AIR 1.0.

- Jiang AP, Kennedy DN, Baker JR, Weisskoff R, Tootell RBH, Woods RP, Benson RR, Kwong KK, Brady TJ, Rosen BR, Belliveau JW. Motion detection and correction in functional MR imaging. Human Brain Mapping 1995;3:224-235.

Validation of AIR 1.0 in the context of fMRI.

- Black KJ, Videen TO, Perlmutter JS. A metric for testing the accuracy of cross-modality image registration: Validation and application.Journal of Computer Assisted Tomography 1996;20:855-861.

AIR MRI-PET registration validation in monkeys

- West J, Fitzpatrick JM, Wang MY, Dawant BM, Maurer CR, Kessler RM, Maciunas RJ, Barillot C, Lemoine D. Collignon A, Maes F, Suetens P, Vandermeulen D, van den Elsen P, Napel S, Sumanaweera TS, Harkness B, Hemler PF, Hill DLG, Hawkes DJ, Studholme C, Maintz JBA, Viergever MA, Malandain G, Pennec X, Noz ME, Maguire GQ, Pollack M, Pelizzari CA, Robb RA, Hanson D, Woods RP. Comparison and evaluation of retrospective intermodality brain image registration techniques. Journal of Computer Assisted Tomography 1997;21:554-566.

AIR 1.0's MRI-PET registration technique in a comparative blinded study.

- Imran MB, Kawashima R, Sat K, Kinomura S, Ito H, Koyama M, Goto R, Ono S, Yoshioka S, Fukuda H. Mean regional cerebral blood flow images of normal subjects using technetium-99m-HMPAO by automated image registration. Journal of Nuclear Medicine 1998;39:203-207.

Use of AIR for intersubject registration of HMPAO-SPECT images.

F.37 ACKNOWLEDGMENTS

Scott T. Grafton, Simon R. Cherry and John C. Mazziotta have been instrumental throughout the development of AIR providing support, encouragement, data, constructive recommendations and feedback. John D.G. Watson, Colin J. Holmes, and Nancy L. Sicotte have been active collaborators in the validation of subsequent versions.

Feedback and encouragement from groups using earlier versions of AIR contributed to the decision to make the software more widely available. John Watson, Ralph Myers, Richard Frackowiak, Jon Heather, Mark Mintun, Tom Nichols, Joel Lee, Tom Zeffiro and Tom Grabowski were especially helpful and

supportive. Aiping Jiang, David Kennedy and their collaborators have played an active role in validating the use of AIR for fMRI data.

The decision to incorporate sinc interpolation in AIR was a direct consequence of Joe Hajnal's important work in this area. Chirp-z interpolation would not have been implemented had Robert Cox not brought this technique to my attention.

Bugs have been identified by Marco Iacoboni, Darren Emge, Kate Fissel, Tom Grabowski, Mark Evans, Greg Ward and others whose names I have misplaced.

Mark Evans provided detailed and helpful suggestions that should make AIR version 3.05 and later portable to PC's and Macintoshes without needing to modify the souce code.

At UCLA, Eric Behnke, George Branch, Rick Cai,Robert Knowlton, Michel Levesque, Larry Pang, Michael Phelps, Ron Sumida, Arthur Toga, Charles Wilson, and Jingxi Zhang all contributed time or resources helpful in the validation of the software.

Mirence Sibomana supports a CTI ECAT wrapper for AIR.

CTI (Knoxville, TN) provided use of the Sun SPARCstation used for software development.

Validation of the software has been supported by Department of Energy cooperative agreement DE-FC03-87ER 60615, National Institute of Mental Health grant R01-MH37916 and NIH-NINDS grant P01-NS15654 and NIH-NINDS grant 1 K08 NS01646. Salary support during the initial rewriting of the software and documentation for widespread distribution was provided by a Fellowship from the Dana Foundation as a Dana Scholar in Neurosciences, and current salary support is provided by NIH-NINDS grant 1 K08 NS01646. Continued support for software development and distribution are provided by The Ahmanson Foundation, the Pierson-Lovelace Foundation and the Brain Mapping Medical Research Organization.