Align3_TP Manual

J. Anthony Parker, MD PhD
Beth Israel Deaconess Medical Center
Boston, MA
J.A.Parker@IEEE.org
Revised: 26 November 2004

General

ImageJ is a highly versatile image processing program written by Wayne Rasband in java which will "run anywhere". It has a particularly outstanding application programming interface (API).

Align3_TP is an ImageJ plugin which allows the user to align two image stacks (or two images), by successively aligning 2-dimensional slices, including oblique slices. (The alignment is defined by an Affine transform.) Output is either an aligned stack or a fused, aligned stack. The alignment may be applied to a third stack; for example, the transmission data from a PET scan may be aligned with a CT and then that alignment can be applied to the emission data. An advantage of using 2-dimensional operations is speed. Another advantage of using a sequence of operations is that the user can interact, supplying knowledge, during the process. However, that also means the registration process can be time consuming for the user.

Align3_TP treats the stacks a 3-dimensional data, not as a series of 2-dimensional images. It is not intended for operations on a sequence of 2-dimensional images. It will work with two single images.

The Align3_TP commands are described below. For a quick start, you may want to skip to the Instillation and Running Align3_TP sections, and then skim the Alignment and View sections.

Data Transformation

The regnant slice from each stack is displayed in a separate window. (The regnant or reigning slice is the currently selected and displayed slice.) The regnant slice is defined as the z=0 plane of the following transformations:

<table>
<thead>
<tr>
<th>slice1</th>
<th>View{z-scaling{stack1}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>slice2</td>
<td>View{Alignment{z-scaling{stack2}}}</td>
</tr>
</tbody>
</table>

where:

<table>
<thead>
<tr>
<th>View</th>
<th>The displayed slice is defined by an affine transform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>The alignment between the stacks is defined by an affine transform.</td>
</tr>
<tr>
<td>z-Scaling</td>
<td>The sample spacing between slices is often different from the sampling within the axial plane for radiological data. Scaling in the slice (z) direction facilitates visualization of the data, but is optional for output data.</td>
</tr>
</tbody>
</table>

An affine transformation is a linear transformation defined by a 3x3 matrix followed by a translation. An affine transformation allows rotation, mirroring, scaling, shearing, and translation in 3-dimensions. (In this
plugin, 3x3 matrix plus 3 offsets is used instead of the somewhat more common 4x4 matrix formulation.)

**Color**

Align3_TP uses short (16-bit integer) or float (32-bit floating point) data internally. The input stacks may be any format, but the slices and output stack(s) are in short or float format. RGB color is not supported by this plugin. Note, however, that an output stack may be colorized (e.g. select a LUT and then convert to RGB) and the colorized output stack can then be combined with another stack to produce a color coded, fused stack. Also see Fusion_TP, which will fuse and colorize two stacks.

**Installation**

The ImageJ home page for instructions for installing ImageJ. Align3_TP can be installed by either unzipping the class files in the plugins folder or by recompiling the java source. Recompiling should only be necessary if changes are made to the code (not recommended). To recompile the source, the "Align Stacks" folder must be in the ImageJ classpath.

ImageJ allows plugins to be "installed" in a menu with an "argument", use "Plugins/Shortcuts/Install Plugin...". Currently, Align3_TP defaults to a right handed axial stack which goes from caudal to cephalic, where these words have meaning in a radiological context. To change to a left handed axial stack where the images go from cephalic to caudal, install Align3_TP with an argument which includes "left". Install with an argument which includes "about" to include the about message. Install with an argument which includes "external" to include options which allow communication with an external registration program (see 6 in the Alignment section). Install with an argument which includes "automatic" to switch to automatic mode (see 7 in the Alignment section).

**Running Align3_TP**

Two stacks (or images) need to be opened prior to selecting Align3_TP from the plugin menu. The user selects the stacks (or images) to be used.

The currently slice from each stack will be displayed in a separate window. On start up, the Align3_TP frame is displayed:
While this Align3_TP frame is displayed, other ImageJ commands can be performed. For example, the slices or the fused slice may be saved.

Some of the buttons affect the alignment of the two stacks, see Alignment below, changing the orientation of the second stack with respect to the first. Other buttons affect the view, see View below; the view for both stacks is changed at the same time. The basic idea is that the user selects a view, aligns the images in this view, and then selects another view, aligns, etc. until he/she is satisfied with the 3-dimensional alignment.

**Start over**

The "Start over" option will clear the slice windows and the fused window (if any), discard any view, alignment, or stack scaling, allow selection of new stacks (images), and rebuild the slice windows using the current slice from the input stacks.

**Layout images**

The "Layout images" option places the two stacks at the bottom left of the screen and displays the regnant slice from each of the stacks above them. The fused slice (if any) will be displayed at the top left of the screen.
Undo

The "Undo" option allows the user to pick any previous alignment and view.

History in/out

"History in/out" restores or saves the sequence of operations. After retrieving a saved history, the images will be aligned the same way as when the history was saved. The "Undo" options from the previous session will also be available.

Alignment

The alignment between the stacks is defined by an affine transform. This transformation is modified by the following options.
1. Align slices

The "Align slices" option shows a fused image of slices 1 and 2. A panel allows the user to input alignment parameters until the images are satisfactorily aligned.

The "OK" button shows a new fused image and returns to the "Alignment Parameters" display. The "Cancel" button is a method of leaving "Align slices". ("Cancel" is really a "Done" button, but ImageJ does not provide an option to change the name of this button.) "Cancel" only cancels the changes made since the last "OK". Alternately, checking "Done with alignment" will leave "Align slices", implementing the current entries.

Rotate and mirror affect the alignment between Slice 1 and Slice 2. The view for both slices can be changed by the "Rotate view", "Mirror horizontal", and "Mirror vertical" options, see below.

2. Match ROIs
If square or oval ROIs are drawn on each image, then the "Match ROIs" option will scale and shift slice 2 so that the ROIs are the same size in the same position. "Match ROIs" is often a convenient first operation to perform.

3. Shear horizontal and vertical

If straight line ROIs are drawn on each image, then the "Match ROIs" option will rotate slice 2 so that the two lines would be in the same orientation.

If a straight line ROI is drawn on slice 2, then the shear operations will shear the image so that the line would be vertical or horizontal respectively. If a straight line ROI is drawn of both slices, then the shear is
the angle between the two lines.

4. Mouse translate, rotate, resize, and cancel

After selecting "Mouse translate", "Mouse rotate", or "Mouse resize", the mouse can be dragged on the slices or on the fused slice to perform these operations. "Dragging" is performed by a click and hold on the mouse button. During dragging, the fused image is updated to show the effect of the operation. For large images on slow processors, updating can be slow. When the mouse is released, slice 2 is updated, and the mouse operation is exited. After a mouse operation has been selected, but before the mouse is clicked on an image, the operation can be canceled with "Mouse cancel".

5. Slice registration & Volume registration

Following is a terse description of the automatic registration options. Each of the algorithms is described more precisely in comments at the beginning of the code. Of course, the code provides the most exact, if often opaque description.

"Slice registration" performs automatic 2-dimensional registration of the regnant slices using either all of the overlapping data or only the overlapping data within the ROI defined on slice 1. "Volume registration" performs automatic 3-dimensional registration either on the whole volume or on the regnant 3D region. "Volume registration" using the whole volume can be too lengthy to be practical for moderate sized images.

"Volume registration" is different from all of the other options in Align3_TP in that it uses the whole 3-dimensional datasets instead of the 2-dimensional regnant slices.

"Function = xxxx" shows the value of the similarity function during optimization. The final test position is often not the optimal value of the similarity function, so the last number displayed will generally not be the largest number. The number of iterations of the optimization algorithm and the total number of evaluations of the similarity function is shown upon completion. "Slice registration" and "Volume registration" are in evolution, consequently they are littered with options.
Similarity functions include correlation coefficient, sum of absolute differences, and [normalized] mutual information. Transformations include various types of affine (non-warping) operations. The Optimization algorithms include Powell's algorithm, a parabolic fit to random points, simulated annealing, or no operation (useful for plotting). Interpolation methods include nearest neighbor, tri-linear and a random jitter.

**Similarity functions:** The correlation coefficient is defined as $\frac{E\{(f-m_f)(g-m_g)\}}{\sqrt{E\{(f-m_f)^2\}E\{(g-m_g)^2\}}}$, where $f$ and $g$ are the pixel values two images (F and G), $E\{}$ is the average over the pixels in the image, and $m_f$ and $m_g$ are the respective mean values. The correlation coefficient is just a cross correlation normalized for the power in the images. The sum of absolute differences is defined as $1-\sum|f-m_f|-|g-m_g|/\sum|f-m_f|+|g-m_g|$, where sum{} is the sum over the pixels in the image.

Both the correlation coefficient and the sum of absolute differences assume that the two images are on the same scale. White objects in F correspond to white objects in G, and black objects in F correspond to black objects in G. These two similarity functions are most appropriate for aligning images from the same modality, e.g. CT to CT. The following example shows a CT and a low resolution transmission map.
Normalized mutual information is defined as $I(F,G)/H(F,G)$, where $I(F,G)$ is the mutual information of two images, $F$ and $G$, and $H(F,G)$ is the joint entropy of the two images. The mutual information is defined as $\sum pdf(f,g) \log \frac{pdf(f,g)}{pdf(f)\cdot pdf(g)}$, where $pdf(f,g)$ is a two-dimensional histogram of pixel values in images $F$ and $G$, and $pdf(f)$ and $pdf(g)$ are the marginal one-dimensional histograms of the pixel values in the respective images. The joint entropy is defined as $\sum pdf(f,g) \log \frac{1}{pdf(f,g)}$.

The advantage of mutual information is that the two images do not need to be on the same scale. White objects in $F$ can correspond to back objects in $G$, etc. The requirement is only that regions of a single intensity in $F$ correspond to regions which have a single intensity in $G$ (or a small number of intensities in $G$). The two intensity values do not need to be the same, rather the region with a single intensity value in $F$ needs to correspond to a region in $G$ which has a single intensity value (or a small number of intensities). Mutual information is particularly useful for multimodality comparison, e.g. CT to MRI. Random jitter interpolation is most appropriate for mutual information. The following example shows the first image from mri-stack.tif and the same image which was color coded, converted to RGB, converted to 8-bit color, inverted, and redisplayed in black and white. Although the intensities have been altered, the same regions are recognizable; this feature is the key feature for the mutual information approach.

**Transformation:** Various affine (non-warping) transformations are allowed. Limiting the number of parameters which are optimized increases the probability of success. I have experimented most with "Translate", "Scale", and "Rotate". I have experimented much less (and much less successfully) with combinations of operations. "Shear" has not been adequately debugged.

**Optimization algorithm:** The optimization algorithm is a procedure for changing the transformation.
Based on the value of the similarity function, a sequence of transformations is tested in an attempt to find the transformation which results in maximizing the similarity function. Powell's algorithm and simulated annealing are taken from *Numerical Recipes*.

Powell's algorithm does a sequence of 1-dimensional searches for a maximum in the similarity function.

Simulated annealing uses a downhill simplex algorithm which has been modified so that it will occasionally go up hill. Up hill moves occur with a probability determined by the simulated annealing algorithm. The idea of allowing up hill moves is to allow the algorithm to get out of local minima. As the optimization progresses, a "temperature" decreases, and this "temperature" determines how likely it is for the algorithm to go up a hill based on the ratio of the hill size to the "temperature". It jumps over big hills initially looking for deep valleys; subsequently, it jumps over lower hills finding the deepest point within the valley it found during the earlier more vigorous jumping. The downhill simplex algorithm requires optimization of at least two parameters, so this algorithm cannot be used if only a single parameter is being optimized.

The parabolic fit algorithm is just a parabolic fit to a set of samples at random points over the rage specified. The parabolic fit is performed for each parameter separately. It often provides the most accurate sub-pixel registration. It tends to be least affected by the method used for interpolation.

"No operation" may be used to plot without first running an optimization.

**Interpolation:** After transformation, the new pixel locations will not in general line up with the old pixel locations. Nearest neighbor interpolation uses the value from the pixel which is nearest to the position of the new pixel. Tri-linear interpolations uses an average of the surround pixel values weighted by the distance of the new location to the old locations. Random jitter uses the value of one of the surrounding pixel values chosen at random depending upon the distance to the original pixel locations. The random jitter is uniform over one pixel spacing.

The similarity functions tend to have local maxima where the matrix points are aligned, especially tri-linear interpolation. Interpolation using random jitter tends to de-emphasize these maxima. Random jitter is particularly appropriate for the mutual information similarity function. Slice 2 is redisplayed using the selected interpolation method in order to show the effects of the interpolation. The redisplayed image is most useful for showing the effects of random jitter. (This interpolation is only used for the first redisplay.)

**Parameters:** For different options, various parameters need to be entered. Only certain of the parameters are used depending upon the options selected. The initials connect the options with the parameters. It's a bit quicker, if somewhat more confusing to have all the options on the initial dialog rather than having a sequence of dialogs.

If "CC: Std for smoothing" is non-zero when using the correlation coefficient similarity function, then the second stacks is smoothed with a Gaussian defined by this standard deviation. Since this is a 3-dimensional operation, it can take time (and memory). Smoothing has been described as a method of overcoming local maxima caused by interpolation; however, in my hands it seems not to get rid of the local maxima, merely to make the local maxima smaller. Making the maxima smaller doesn't really help with Powell's algorithm, although presumably it could help with simulated annealing.

"PW, SA: stopping rule (pixels)" gives a stopping rule for Powell's algorithm and simulated annealing. When changes are less than the number of pixels given by this parameter, the algorithm stops. In *Numerical Recipes*, stopping is defined by machine precision. Since interpolation limits registration accuracy to something on the order of a couple of tenths of a pixel, I have added a stopping rule in terms of pixels.
"PW, SA: minimum overlap (%)" gives the minimum amount of overlap between images. This minimum and the change penalty affect the similarity function not the optimization algorithms, so it is still possible to get to some run away situations.

"PF: range" and "PF: samples" give the range and the number of samples used in the parabolic fit optimization algorithm. A number of samples is chosen at random over a particular range in pixels. The values of the similarity function are fit to a parabola and the maximum is determined.

"SA: temperature" is the starting temperature for the simulated annealing algorithm. This temperature should be similar to the changes in the value of the similarity function.

"SA: decrease per step (0,1)" is fraction by which the temperature is decreased during each step of the simulated annealing algorithm. This parameter and the next determine how fast the temperature will decrease.

"SA: moves per step" is the number of steps the simulated annealing algorithm makes between changes in the temperature. It might be appropriate to increase this parameter or decrease the former parameter when optimizing for a larger number of transformation parameters.

**Show moves:** Show moves is for debugging.

**Change penalty:** The similarity function can be penalized for large changes in order to keep the optimization form going to unreasonable solutions. The penalty is defined as

\[
\text{halfPenalty}^2/(\text{distance}^2+\text{halfPenalty}^2),
\]

where "halfPenalty" is the distance where the penalty is 1/2, and "distance" is the distance in pixels of the test movement. The penalty, which varies from 1 to 0 as the distance becomes larger, is multiplied by the similarity function. (The optimization parameters are scaled so that a value of 1 corresponds to about a change in the image of 1 pixel.) The change penalty is most effective (changes the most) near the halfPenalty point. If the algorithm manages to get to a distance which is large with respect to halfPenalty, run away can occur.

**Plot:** Plotting will show a plot of the similarity function along each of the parameters which is being optimized. It can sometimes be helpful in identifying a local maximum.

6. **External**

If Align3_TP is installed with argument "External" options appear which allow storing and retrieving data. The purpose of this option is to allow an external registration program to be used. For example, the images could be roughly aligned with this plugin, precisely aligned with an external program, and then the alignment could be viewed with this plugin. This option is not further described in this manual.

7. **Automatic**

Align3_TP is intended to be run in an interactive mode by an imaging expert. However, if Align3_TP is installed with argument "Automatic", it will run in an automatic mode. In automatic mode, it will follow a script contained in "plugins/Align Stacks/Align3_TP.script". Command templates are found in "plugins/Align Stacks/Align3_TP_Template.script". The template file tersely describes the script syntax, and which commands are available for scripting. The script files need to be fairly accurate, and there is
little error checking. This option has not been extensively debugged.

Show 3D region & Set 3D region

"Show 3D region" and "Set 3D region" set and show two dimensions of a 3D rectangular region of interest using an ImageJ ROI. They work only when the view is axial, coronal, or sagittal, e.g. lined up with the data matrix. "Show 3D region" and "Set 3D region" are used with volume registration and with the external registration option.

View

The slice which is displayed to the user is defined by an affine transform. Several options affect the view.

1. Resample along line

The straight line tool is used to define a plane perpendicular to the display plane on either slice.

The "Resample along line" option will change the view to a plane along the straight line which is perpendicular to the current slice.

2. Axial, Coronal, and Sagital
If the initial stack is a stack of axial slices, "Axial", "Coronal", and "Sagittal" will change the view to these orientations. These views are meaningful in a radiology context. They may make less sense in other applications.

3. Rotate view

The "Rotate" option rotates the view by a specified angle. This option affects the "view", i.e. both slices. Slice 2 can be separately rotated using the "Align slices" option, see above.

4. Mirror horizontal and vertical

The "Mirror horizontal" and "Mirror vertical" options invert the respective axes. These options affect the "view", i.e. both slices. Slice 2 can be separately mirrored using the "Align slices" option. If the initial stack is a stack of axial slices from caudal to cephalic, then the slices should generally be displayed in standard radiological orientation. For a "left hand" stack, where the axial images go from cephalic to caudal, the z axis will need to be mirrored once.

5. Increment and decrement slice

The "Increment slice" and "Decrement slice" options move the view one pixel perpendicular to the display plane.

6. Goto slice and Goto again

The "Goto slice" option allows incrementing or decrementing the slice by a number of pixels, or allows a particular slice to be selected. The "Goto again" repeats the last "Goto slice" operation, or the default operation if "Goto slice" has not been performed.

Scale stacks (z)

Sampling in the slice direction is often different from the in plane sampling for radiological data. Scaling in the slice (z) direction facilitates visualization of the data, but is optional for output data.

Show Affine

The "Show Affine" option prints out the alignment and view affine transformations in the ImageJ window. The view transforms for both slices are the same. The alignment transform for slice 1 is not used.

Output

Individual slices and fused slices may be saved with ImageJ commands. The "Output" options lets the user make a new stack from stack 1, stack 2, or the current stack. Stack1 and stack 2 or stack1 and the current stack may be fused using the operation defined in the "Align stacks" option. The "current stack" is the stack which is highlighted. The "current stack" may be a third stack which can be read into ImageJ at any time before selecting the output option. "using alignment" will make an axial stack which has been transformed by the alignment of stack 2 to stack 1. "using alignment & view" will make an aligned stack using the current slice view. "Interpolate" and "Blitter op" have the same values in Align slices and Output.
options for output which make the most sense can be listed in the ImageJ window by selecting the "Output help" option.

The output stack can be saved, combined with other stacks, etc. using all the ImageJ functionality.