1 Program imreg

1.1 Purpose

This program takes as input a sequence of (2D) images and a base image, and produces as output a sequence of images registered to the base image. Note that the images must be acquired in the same modality — there is no allowance for having different types of tissue-to-grayscale mapping functions between the base image and the sequential images. Besides the output images, imreg also reports the Δx, Δy, and $\Delta\theta$ values needed for each registered image (in pixels, pixels, and degrees, respectively).

By the way, imreg (and imrotate) use bicubic interpolation when resampling the images to the new grid system. Imreg uses the same routines as fim2 for registration, which are designed only to deal with small displacements (at most 3-4 pixels).

1.2 Usage

imreg [options] base_image image_sequence ...

- Registers each 2D image in 'image_sequence' to 'base_image'.
- If 'base_image' = '+AVER', will compute the base image as the average of the images in 'image_sequence'.
- If 'base_image' = '+count', will use the count-th image in the sequence as the base image. Here, count is 1,2,3, ...

1.3 Output Options

-nowrite Don't write outputs, just print progress reports.

-prefix pname	Default pname	=	'reg.'
-suffix sname	Default sname	=	nothing at all
-start si	Default si	=	1
-step ss	Default ss	=	1

The output files will be named in the format

'pname.index.sname'

where 'pname' and 'sname' are strings given by the first 2 options. 'index' is a number, given by 'si+(i-1)*ss' for the i-th output file, for i=1,2,...

-flim Write output in mrilib floating point format (which can be converted to shorts using program ftosh).

- Default is to write images in format of first input file in the image_sequence.

-quiet Don't write progress report messages.

-debug Write lots of debugging output!

-dprefix dname Write files 'dname'.dx, 'dname'.dy, 'dname'.phi for use in time series analysis.

1.4 Alignment Algorithms

-bilinear Uses bilinear interpolation during the iterative adjustment procedure, rather than the default bicubic interpolation. NOT RECOMMENDED!

-modes c f r Uses interpolation modes 'c', 'f', and 'r' during the coarse, fine, and registration phases of the algorithm, respectively. The modes can be selected from 'bilinear', 'bicubic', and 'Fourier'. The default is '-modes bicubic bicubic bicubic'.

-mlcF Equivalent to '-modes bilinear bicubic Fourier'.

-wtim filename Uses the image in 'filename' as a weighting factor for each voxel (the larger the value the more importance is given to that voxel).

-dfspace[**:0**] Uses the 'iterated differential spatial' method to align the images. The optional :0 indicates to skip the iteration of the method, and to use the simpler linear differential spatial alignment method. ACCURACY: displacments of at most a few pixels.
Note: This is the default method (without the :0).

The next two options are used to play with the -dfspace algorithm, which has a 'coarse' fit phase and a 'fine' fit phase:

-fine blur dxy dphi Set the 3 'fine' fit parameters:

blur	=	FWHM of image blur prior to registration, in pixels [must be > 0];
dxy	=	convergence tolerance for translations, in pixels;
dphi	=	convergence tolerance for rotations, in degrees.

-nofine Turn off the 'fine' fit algorithm. By default, the algorithm is on,
with parameters 1.0, 0.07, 0.21.

1.5 How Program "imreg" Works

Registration is accomplished by minimizing the error functional

$$E(I,J) = \sum_{x,y} \left[I(x,y) - J(T(u,v,h)(x,y)) \right]^2$$

where $J(x,y)$ is the "base" image, $I(x,y)$ is the image to be registered to it, $T(u,v,h)$ is the transformation with shift parameters (u,v) and rotation angle h. In practice, $J(T(x,y))$ is expanded in a Taylor series in (u,v,h) to be $J0(x,y)+u*Ju(x,y)+v*Jv(x,y)+h*Jh(x,y)$. The minimization is then a linear least squares problem. The (u,v,h) that minimizes E is found. At this point, $I(x,y)$ is transformed with $T(-u,-v,-h)$ to bring it closer to $J(x,y)$ (bicubic interpolation is used for resampling I). Then the minimization is repeated – that is, simple gradient descent is used to minimize E.

Actually, this procedure is performed with a $J(x,y)$ that has been smoothed with a Gaussian filter with FWHM=4 pixels. In this way, the effects of pixels a little farther away than nearest neighbors are included in the derivatives, and displacements of up to 2-3 pixels can be detected.

When the FWHM=4 pixel smoothing iteration converges, it is then repeated with $J(x,y)$ smoothed with FWHM=1.0 pixels. Typically, 2-4 iterations are required to converge in the first step, and 1-2 in the second, if the displacements are larger than 1/2 pixel.

The programs imreg and fim2 both use the routines in "mri_align.c", if you wish to see the details. The Gaussian smoothing and the differentiation of $J(x,y)$ are both done with FFTs.

One reference to a similar method is:

> Michael Irani and Shmuel Peleg, Improving Resolution by Image Registration. CVGIP (Computer Vision, Graphics, and Image Processing), Vol 53, pp. 231-239 (1991).

There are references in this paper to earlier works.

I strongly recommend that you use imreg to register image time series and look at the results to make sure that the program is doing a good job for you. Note that this method will only work for registration of similar images displaced by only a few pixels. It cannot be used to register PET and MRI, for example, or to register FSE and EPI.

Some further notes on motion and FMRI in general can be found at

http://www.biophysics.mcw.edu/BRI-people/rwcox/regnotes.ps

– this is a 780 K PostScript file, 17 pages long.

1.6 Examples

Example 1. Suppose that the user has a time series of 2D images, for slice z = 8, in files fred08.0001, fred08.0002, ..., fred08.0068. The command to register these images to the *average* of the 68 images is given by:

```
imreg -prefix fredreg -dprefix dname +aver fred*
```

to which the computer responds with:

```
– will set base image to AVER
– computing AVER image now
– beginning alignment
– mri_align_dfspace.............
– registering.............
– image fredreg.0001: dx = 0.057 dy = 0.104 phi = -0.171
– image fredreg.0002: dx = 0.104 dy = 0.010 phi = 0.079
– image fredreg.0003: dx = 0.054 dy = 0.010 phi = 0.131
– image fredreg.0004: dx = 0.017 dy = 0.028 phi = 0.068
⋮
etc.
⋮
– image fredreg.0065: dx = 0.051 dy = -0.013 phi = -0.072
– image fredreg.0066: dx = -0.000 dy = -0.007 phi = -0.087
– image fredreg.0067: dx = -0.001 dy = 0.029 phi = -0.141
– image fredreg.0068: dx = 0.031 dy = 0.023 phi = -0.183
– MAX: dx = 0.104 dy = 0.104 phi = -0.183
```

At this point, the registered 2D images are contained in files fredreg.0001 through fredreg.0068. Note that for each of the registered images, the computer printed out the "adjustment": dx and dy, in pixels, and the rotation angle phi, in degrees. If the user wishes to study the time series of adjustments, the `-dprefix dname` command can be used as shown above. For example, file dname.dx contains the dx values:

```
0.057
0.104
0.054
0.017
⋮
etc.
⋮
0.051
-0.000
-0.001
0.031
```

Similarly for files dname.dy and dname.phi.