Registration Practical

Practical Overview

In this practical you will explore each of the registration steps within a standard two-step registration for functional images. We will first learn to use the registration tools within the FEAT GUI. Then we will see how to apply and invert transformations. Being able to achieve precise registrations is CRUCIAL for structural, functional and diffusion image analysis. If registrations are not accurate, further statistics at a structural or group level will not be accurate.

Core Content:

- **1: Two-stage Registration and Unwarping with FEAT**
  Register functional EPI images to the individual structural image and to standard space using the FEAT GUI. This includes fieldmap-based unwarping and the preparation of these fieldmap scans.Register functional EPI images to the individual structural image and to standard space using the FEAT GUI. This includes fieldmap-based unwarping and the preparation of these fieldmap scans.
- **2: Applying and Inverting Transformations**
  Calculate and apply transforms and inverses from linear and non-linear registration (as output from FEAT). This provides experience in transforming masks (or other images) between the different "image spaces" (functional, structural, standard). The same principles apply for registration in diffusion imaging, and the similarities and differences are highlighted here. Calculate and apply transforms and inverses for both FLIRT and FNIRT. This provides experience in transforming ordinary images and masks between the different "image spaces" (functional, structural, standard)

Two-stage Registration and Unwarping with FEAT

We have created a step-by-step registration document that has been uploaded onto the FSL wiki for your reference. This may help you through this practical.

1. Take a look inside the data directory (`ls registration`)
2. You should see directory for subject 001
3. There is an additional directory `registration/001_new_reg` which we will be using in this practical, to replicate and manipulate the registrations provided for subject 001

```
cd ~/fsl_course_data/registration/001_new_reg
```

This directory contains the following images:

1. A structural scan: `STRUCT.nii.gz`
2. A functional scan: `FUNCT.nii.gz`
3. A magnitude fieldmap image: `FMAP_MAG.nii.gz`
4. A phase fieldmap image: `FMAP_PHASE.nii.gz`
5. A text file containing the fieldmap information: `FMAPS.txt`

The objective of this practical is to become familiar with how to perform and evaluate the results of registration in the FEAT GUI (the main functional analysis GUI). This involves two-stage registration and fieldmap-based unwarping, and requires the images to be suitably prepared. Registration in other FSL GUIs (e.g. for ICA or diffusion analysis) works very similarly.

Preparing the structural image

We need to perform brain extraction on the structural image prior to using it for registration (as explained in the lecture). To do this run BET on the image `struct.nii.gz`, and save the result as `STRUCT_brain.nii.gz`. Check the results with `fslview`.

Fieldmap images

There are two images that the Siemens scanner saves from a fieldmap sequence. These are (in this case, renamed nicely for you):

```
FMAP_MAG.nii.gz
FMAP_PHASE.nii.gz
```

where the first represents the standard magnitude parts of the images with the different echo times, and the last one represents the difference of the phase parts of the two images. Have a quick look at these with fslview. We will need to use both magnitude and phase images.

Processing the fieldmap magnitude image

We will start by brain extracting the first magnitude image. Run brain extraction on the magnitude image and save the result as `FMAP_MAG_brain.nii.gz`. We will then erode this image (shaving off one voxel from all edges) as this image contains noisy partial volume voxels in the phase difference image near the edge of the brain (you will see this below, whereas in practice you would look first in order to decide whether to erode or not). To do the erosion we run:

```
        fslmaths FMAP_MAG_brain -ero FMAP_MAG_brain_ero
```

This performs an erosion operation (stripping one voxel from the edge) with the general tool `fslmaths` which acts as an image calculator, taking input images and performing operations on them and then saving them as a new image (the last name specified). Check what the results look like in `fslview` and add (to the viewer) the phase (`FMAP_PHASE.nii.gz`) to see why we needed to erode the brain in order to avoid the noisy voxels at the edge. Note that the "lost" voxels will have fieldmap values filled in by extrapolating from the nearest voxel inside the mask, which is quite accurate for fieldmaps as the field variations are quite smooth (you cannot see this in these images as it happens in the later processing).

Processing the fieldmap phase difference image

Here we need to take this raw scanner output, which is scaled in a strange way (0 to 360 degrees are mapped to 0 to 4096), and convert it into radians per second image (this is equivalent to an image in Hz multiplied by 2*pi). For this we need the phase difference image, the brain extracted (and eroded) magnitude image and the difference in the echo times of the fieldmap acquisitions. This latter value is 2.46ms and can be found in the text file `FMAPS.txt`, which is conveniently given here but will not exist normally. Therefore it is important to record this echo time difference when you scan (your scanner operator will be able to give you the value, and although it can usually be determined later on, it is much easier to record it at the time when the scanner operator is present).

Armed with this information, all we need to do is run the GUI called `Fsl_prepare_fieldmap`. Note that these images come from a Siemens scanner. Set up all the information required in the GUI, (phase image is `FMAP_PHASE.nii.gz` and the brain extracted magnitude image is `FMAP_MAG_brain_ero.nii.gz`) calling the output `FMAP_RADS`, and press `Go`. View the output with `fslview` and check that most of the brain has small values (less than 200 rad/s) while in the inferior frontal and temporal areas the values are larger (either large and positive or large and negative).

Using the FEAT GUI

With the fieldmap processed and the structural image brain extracted we are now ready to use the FEAT GUI for registration. Note that the ‘unwarping’ of images using fieldmaps is done in the ‘Pre-stats’ tab of the FEAT GUI.

Start the FEAT GUI by typing `Feat` in the terminal. From the drop-down list in the top right corner select ‘Preprocessing’. We now need to set up the GUI to run our registration with unwarping.

We will start with the main `Data` tab. To begin with click on the Select 4D data button and select the image `FUNCT.nii.gz`. Once this is done, click on the file browser for the Output directory and make sure the directory is set to somewhere in your account (it would normally default to something like `~/fsl_course_data/registration/001_new_reg/feat_example`). Then, in the Selection box at the bottom type ‘`001_reg`’ and press `OK`.

Now go to the `Pre-stats` tab and click the B0 unwarping button. Note that all other parts of this tab (e.g. motion correction) are set to their default values and we will leave them with these settings. Now go through and populate the relevant parts: using FMAP_RADS for the "Fieldmap" (but select this using the file browser that the folder icon opens); `FMAP_MAG_brain_ero` for the "Fieldmap mag"; a value of 0.50 ms for "Effective EPI echo spacing" (taken from the FMAPS.txt file here, but in general you need to take note of this value when scanning - ask your operator - and if you are using parallel acceleration then it needs to be divided by the acceleration factor); 28ms for "EPI TE"; "-y" for "Unwarp direction" (again your operator can tell you whether the unwarp direction, or phase encode direction, is x, y or z, but whether it is + or - needs to be determined, currently, by trial and error, which has already been done here); and leave the "% Signal loss threshold" at the default value of 10%.

Go to the `Registration` tab and click on the "Main structural image" button. Open the file browser and select `STRUCT_brain`. Make sure the options underneath are set to Normal search and BBR. In the Standard space box, reselect the image `MNI152_T1_2mm_brain` in the correct directory. (This reference image is part of FSL. In general, to find out where FSL is installed, type "echo $FSLDIR" into a terminal. The standard space reference images are in data/standard inside that directory.) We will not select the "Nonlinear" button in the Standard space section for this part of the practical in order to save time, but you normally would use this setting as it gives the best results.

OK, that's everything we need to register the functional image to standard space. So double-check that the "Pre-stats" and "Registration" tabs look correct and when you are happy press the Go button at the bottom. This should start up a web browser showing the progress of FEAT - although it may take a minute for this to appear.Now go back to the FEAT GUI, and we are going to run a comparison registration WITHOUT FIELDMAP UNWARPING. So go to the Pre-stats tab and de-select the B0 unwarping button. In the Data tab, re-name the output to something like ‘`001_reg_no_fmap`’. Everything else stays the same, and once you are happy with all the setting press the Go button again.

While you wait

The FEAT job will take about 10 minutes to finish. In the meantime we can have a look at the registrations provided in the 001 subject directory.

```
cd ~/fsl_course_data/registration/001/FUNCT_1stLevel.feat/reg/
```

Open the webpage report for the registration:

```
firefox ~/fsl_course_data/registration/001/FUNCT_1stLevel.feat/report_reg.html &
```

Look carefully at the results of the registrations in the web page report and in the "Unwarping" page for the cases where fieldmap-based correction was run. Do these registrations seem accurate to you? Note that you should not trust borders with signal loss areas as these are not true anatomical boundaries but artificial borders.

It is also highly recommended to use `fslview` to look in more detail. We can look at each of the two registration steps separately (functional to structural, and structural to standard), but remember when these two steps are combined to produce a functional to standard transformation, the functional image is only resampled ONCE into standard space. Let's first look at the initial registration step (functional to structural) in fslview. Load the structural image into `fslview` using the following command:

```
fslview highres.nii.gz &
```

Note: this `highres.nii.gz` image is created in the `reg` folder by FEAT from the T1 structural image we specified.

Using the `fslview` GUI, add the `example_func2highres.nii.gz` image ("`example_func`" is a distortion-corrected example volume from the EPI/functional data, and "`example_func2highres`" is the `example_func` image transformed into structural space), and the `example_func2highres_fast_wmedge.nii.gz` (this image shows the location of the white matter edges, as defined by the highres image). Change the colour of this `wmedge` image to "red" by clicking the "i" button in `fslview` when the image is selected in the image list at the bottom of the GUI. Click around the image to see where the registration is particularly good (as the red edges derived from the structural should align with the changes in the greyscale intensities of the functional image). You can flick between images, or use the transparency bar at the bottom of the fslview GUI to see how well these images align. Feel free to look at other images in this "reg" subdirectory or in the "unwarp" subdirectory inside this. For example, the uncorrected registration result is in the file `unwarp/example_func_distorted2highres.nii.gz` and can be viewed separately or added to the fslview session above.

Now open another fslview session from the terminal to view the second registration step (structural to standard).

```
fslview standard.nii.gz &
```

Add the `highres2standard.nii.gz` image, which is the structural image transformed into standard space. In this case non-linear registration (FNIRT) was used after affine linear transformation (FLIRT) for maximum accuracy. Use the same fslview tools to check the registration of the structural image to the standard image.

Leave both of these `fslview` sessions open for the moment, as we are now going to compare the registrations you just ran to these given examples.

Once the FEAT job is finished

Go back to the `fslview` window where you were looking at the first registration step (functional to structural). Now add the `example_func2highres.nii.gz` image from the registration you ran with fieldmap correction (`~/fsl_course_data/registration/001_new_reg/001_reg.feat/reg/example_func2highres.nii.gz`). How does this registration compare to the original? It should be identical, or at least very, very similar.

Now add the registration you ran WITHOUT fieldmaps (~/fsl_course_data/registration/001_new_reg/001_reg_no_fmap.feat/reg/example_func2highres.nii.gz). You may need to rename this image within `fslview` to something like ‘`example_func2highres_no_fmap.nii.gz`’ to avoid confusion. Use the `fslview` tools you practised earlier to compare the registrations with and without fieldmaps

Question 1: What differences can you see between the registrations with and without fieldmaps? HINT: look closely at the prefrontal cortex and brainstem areas. Why do you think these areas would particularly benefit from fieldmap distortion correction?

Now go to the `fslview` window where you were looking at the second registration step (structural to standard). Add the `highres2standard.nii.gz` image from the registration you ran (`~/fsl_course_data/registration/001_new_reg/001_reg.feat/reg/highres2standard.nii.gz`). This registration step was done linearly (using FLIRT) rather than nonlinearly (using FNIRT). Compare the linear and linear+non-linear versions of this step.

Question 2: What differences can you see between the linear and linear+non-linear registrations of the structural to standard brain? Name some anatomical structures that are better aligned with the non-linear registration. Why is the non-linear registration much more accurate than the linear?

Finally, de-select (or remove) the `highres2standard` images within the `fslview` session. Add the combined registration images from the supplied example using both fieldmaps and non-linear registration (`~/fsl_course_data/registration/001/FUNCT_1stLevel.feat/reg/example_func2standard.nii.gz`) and the registration you ran without fieldmaps or non-linear registration (`~/fsl_course_data/registration/001_new_reg/001_reg_no_fmaps.feat/reg/example_func2standard.nii.gz`). Note: You may need to rename these within `fslview` to avoid confusion. The registration without fieldmaps or non-linear reg will be markedly worse than the original registration, as both sub-optimal registration steps have now been concatenated into one step.

Question 3: Why do we not resample the image twice (i.e. first into structural space and then into standard space) when we transform from functional space to standard space?

Applying and Inverting Transformations

```
cd ~/fsl_course_data/registration/001/FUNCT_1stLevel.feat/reg/
```

The objective of this section of the practical is to become familiar with applying transformations, as well as their inverses, to move masks (or images) between different spaces. This is very useful as although FEAT, Featquery and FDT do a lot of this "behind the scenes" for you, there will be many cases when you want to do something beyond the standard options and then you'll need to be able to do this for yourself.

We will be working with the files from the Feat analysis in the previous part of the practical, however what we are doing is not restricted to functional analysis. FDT outputs similar files when analysing diffusion datasets, and the same files (affine transformation matrices and warp fields) are output by the fundamental tools, FLIRT and FNIRT, when doing any structural analyses.

Transformation files

If you look in this reg directory you will see many different files. The crucial ones for registration purposes are the transformation files that come in two different varieties: affine matrices (ending with `.mat`) and warp fields (ending with `_warp.nii.gz`). For now we will not look at the contents of these files (see the exercises below for more about this) but instead we will explain what each of them means.

The naming convention is always from the input space to the reference space (or source to destination if you prefer). For example, the file `highres2standard.mat` is an affine transformation going from the highres (structural) space to the standard (MNI) space. There is also the file `highres2standard_warp.nii.gz` that represents a non-linear warp from highres space to standard space. We have both of these because it is necessary to initialise all non-linear registrations with an affine registration (that gets the head in roughly the right position and scaling to allow the non-linear registration to work well). When you want to use a transformation between these spaces, you would generally go for the warp field (when it exists) and ignore the initial affine registration. Note that the warp fields include the affine transformation as part of them, so you don't need to use both.

There are three main spaces in a FEAT analysis: functional (represented by example_func); structural (represented by highres); and MNI (represented by standard). Various combinations of transformations exist in this directory

(e.g. example_func2highres, example_func2standard, highres2standard) but not all (e.g. standard2highres). Note that in a diffusion analysis, run via FDT, the three spaces are called diff, str and standard, and all combinations of transformations are provided.

One thing that might confuse you is why there exists `example_func2highres_warp.nii.gz` when BBR was run to do the registration of functional to structural images. This is because of the fieldmap-based distortion correction, which is not just a linear (affine) registration and so must be represented as a warp field.

Creating an example mask

Now let's make a mask in the standard (MNI) space so that we can transform it into the other spaces and use it to calculate some ROI quantities. We will do this using fslview. Start by opening the standard image in fslview (i.e. either start fslview and use the menus to open this image or open it directly from the command line with: fslview standard.nii.gz)

Once this is open, go to the menu and select Tools > Toolbars > Atlas Tools. This will open up an area next to the coordinates that gives you details of various structures in the atlases and how they relate to the point in the brain that the cursor is currently on. We can use these atlases to create a mask image (we will choose the left hippocampus from the Harvard-Oxford subcortical atlas, but there are lots of possibilities).

Click on the Structures... button and in the pop-up window (Atlas Inspector) change the displayed atlas to the Harvard-Oxford Subcortical Structural Atlas using the drop-down menu/list. This will then show you a list of the structures in this atlas. Choose the Left Hippocampus (purely an example) and click on the "+" button at the bottom right of the list. You can dismiss the Atlas Inspector window (if you can see the Left Hippocampus in the image list and an appropriately coloured label in the main fslview window).

Have a look at the hippocampus label in fslview, and choose an appropriate "Min" value to make it more specific (e.g. try 50). The values are expressed as percentage values (0 to 100) and relate to the percentage of subjects that had that particular voxel labelled as the hippocampus from the set of individuals that was used to form the atlas.

We will now save the image using the menu File > Save As... and choose an appropriate name. To make life a lot easier later on make sure you remove all spaces from the filename. This is a general rule to stick with, as spaces within filenames will almost always cause problems and are easily avoided.

In the terminal window, check that you can see the newly saved image (e.g. using `ls`). Then we will threshold this image to make a binary mask. We do this using `fslmaths` and your choice of "Min" value from above. For example, to threshold it at a value of 50 and then binarise the result we do:

```
fslmaths LeftHippocampus -thr 50 -bin LeftHippMask
```

Note that you should use your own filenames (i.e. `LeftHippocampus` and `LeftHippMask` are examples). In fslview, load the mask image to make sure the fslmaths command has run correctly.

Technical Aside: the above label image and mask can also be created directly at the command line using `fslroi` and `fslmaths` with the images in `$FSLDIR/data/atlases` and finding the appropriate values from the xml files (but be warned that the numbers in the xml files differ by one from the numbers needed in `fslroi` - see the FSLWiki for more details).

Inverting a transform

We want to transform the mask we just made into the functional space in order to calculate an ROI value (e.g. of a statistical image, or an average timecourse, etc.). To go from the standard space to the functional space we would need to have the transformation `standard2example_func_warp.nii.gz` (or something named like this). In FEAT this does not exist, but in FDT the equivalent is generated automatically, and called `standard2diff_warp.nii.gz`).

As feat does not create the required warp field, we need to create it ourselves from the transformations that it does provide. In this case we do not want to undo the distortion correction (i.e. we want a mask in distortion-corrected functional space). For the structural-to-functional transform we therefore only need `highres2example_func.mat` which was already created by FEAT. We do need to calculate the nonlinear standard-to-structural transform, which can easily be done using the `invwarp` command, as follows:

```
invwarp -w highres2standard_warp -o standard2highres_warp -r highres
```

where the final part (the reference) controls the size (FOV) and resolution of the warp and should generally be the destination space of the new transform. As this warp transforms from standard space to structural (we will stick on the final transformation to functional space below), we specify highres.

Applying a transformation

Now that we have the transformation that we need, we can apply it using the `applywarp` command, by specifying a warp (`-w`). The command you need is (with your own filenames):

```
applywarp -i LeftHippMask -r example_func -o LeftHippMaskFunc -w standard2highres_warp --postmat=highres2example_func.mat
```

which goes from the standard space via the highres space to the example_func space (the last part only needing an rigid-body transformation matrix, as we want to end up in the distortion-corrected functional space, rather than the original distorted one).

Visual Check

Use the command

```
fslview example_func LeftHippMaskFunc &
```

to see the mask on the functional image. Note that the values at the edge of the mask lie between 0 and 1.

Thresholding the Mask

In order to obtain a binary mask (where each voxel has a value of either 0 or 1) we need to threshold and binarise the transformed mask. This is easily done with fslmaths but the threshold used is arbitrary. If a high threshold is chosen (e.g. 0.9) then most edge voxels will be excluded and the mask will be tighter and less likely to include neighbouring structures. This is often desirable when trying to make sure that only the structure of interest is included, but it might end up with the mask being quite small. So sometimes a threshold near 0.5 is preferable, to make a mask of similar size/volume. Or sometimes a mask is needed that does not leave out any of the structure, in which case a low threshold (e.g. 0.1) can be better. In this case we will choose a high threshold in order to get a mask where we are very confident that each voxel in the mask is within the hippocampus. This can be done with:

```
fslmaths LeftHippMaskFunc -thr 0.9 -bin LeftHippMaskFuncBin
```

Load the resulting image into fslview and compare it to the original, pre-thresholded version in the functional space.

Using the Mask

There are many possible ways in which a mask can be used (e.g. getting average statistical values) but as an example will we calculate the average timecourse of the fMRI within this mask. This is done with the command:

```
fslmeants -i ../filtered_func_data -m LeftHippMaskFuncBin
```

which will output a column of numbers, representing the timecourse of the average signal intensity (of the pre-processed fMRI data) within this mask. We won't do anything with this for now, but such calculations can be very useful in all sorts of situations.

The main point of this exercise was to see how to transform your own masks (or images, as the only difference is skipping the thresholding and binarising steps) between spaces. For functional studies you can often do similar things with the tool Featquery but it is not as flexible and generally useful as being able to process things yourself.

Question 4: Why did we want to transform the mask back into distortion-corrected functional space, rather than the original functional space?