HowTo – 7T registration & distortion correction with ANTs

by Carlos A. Gomes

# Packages you will need

If you want to run the master **ANTs\_7T\_reg.py** script, you need to make sure you have installed:

* Python

Usually already pre-installed

* FSL

[https:/fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslInstallation](https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslInstallation)

* ANTs

[https:/github.com/ANTsX/ANTs/wiki/Compiling-ANTs-on-Linux-and-Mac-OS](https://github.com/ANTsX/ANTs/wiki/Compiling-ANTs-on-Linux-and-Mac-OS)

* Convert3D (for c3d\_affine\_tool)

[http:/www.itksnap.org/pmwiki/pmwiki.php?n=Convert3D.Convert3D](http://www.itksnap.org/pmwiki/pmwiki.php?n=Convert3D.Convert3D)

* Matlab

[https:/www.mathworks.com](https://www.mathworks.com/)

* Add to Matlab path:
* SPM

[https:/www.fil.ion.ucl.ac.uk/spm/software/download/](https://www.fil.ion.ucl.ac.uk/spm/software/download/)

* presurfer

[https:/github.com/srikash/presurfer](https://github.com/srikash/presurfer)

* directory containing the script **MPRAGEise.m**
* Matlab Engine API for Python

[https:/www.mathworks.com/help/matlab/matlab\_external/install-the-matlab-engine-for-python.html](https://www.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html)

* Misc ANTs scripts (written by Sri Kashyap)

# Acknowledgments

If you have used either the entire pipeline or only part of it, please add Sriganga Kashyap ([Sriranga.Kashyap@rmp.uhn.ca](mailto:Sriranga.Kashyap@rmp.uhn.ca)) to the list of co-authors in any publication that might result from it.

# Initial steps

The script will initially copy all necessary files to a working directory. This includes the functional data, opposite phase-encoding data, mp2rage uni and inv2 images.

Because all output will be saved in this working directory, ensure that you have sufficient space in the disk as there will be many files generated and stored in this folder.

Then, you will have the option to trim the functional data (i.e., selecting only a few volumes).

We recommend that you *always* run this registration/distortion correction pipeline first on trimmed functional data. The reason for that is because the entire pipeline is using code that is multi-threaded and will, by default, use all available CPUs on your machine. Second, the pipeline will take a long time, particularly if you have multiple functional runs as distortions are estimated/corrected for each functional run and the compute time scales with the resolution/size of the imaging matrix. The reasons for this are the high-resolution , thus, large matrices of the acquired data and the high degree of precision that is needed for the processing. The compute time will inversely scale with the CPU configuration of your server/workstation.

It will be a waste of time to run the script on the full dataset, having waited a few days to finish, only to find that you will need to improve the brain mask, or use a custom initial registration matrix. While it may seem excessive, this is a critical so as to perform QC at the initial stages for high-resolution fMRI data. Once the results look satisfactory, the pipeline can be queued up to run on the entire dataset.

It should be noted that this pipeline is still likely to be faster than using FSL’s TOPUP or AFNI’s 3dQwarp.

# Brain extraction

An important step in the preprocessing of laminar data is to ensure you have a good functional mask to start with. This is because the mask is used to limit the cost-function estimation during registration. This way, the non-brain fluctuations are not considered and registration quality is more optimal for the brain.

You should always try to use the ants script to create the brain masks. However, sometimes the automatic method to generate a brain mask may not be optimal, as dropouts are different from subject to subject.

Fortunately, you can quickly create brain masks very accurately in ITKsnap, when the ants script **sk\_antsBET.sh** provides sub-optimal results.

It’s important to run the N4 bias field correction algorithm, because there will be a bias field in the EPI data and it’s important to flatten it.

Once the N4 corrected data is generated, open ITKsnap:

* Click on the Snake tool, and it will show you a red dashed box
* Drag the dotted lines to resize this box to confine it to whatever FOV you want to generate the mask over. Choose the middle slice, as it has the largest width/length (if you want to cover the whole acquired FoV). Check it in all views (sagittal, coronal,)
* Select “Segment 3D”. It will crop to the FOV you selected.

For EPI images (3D-EPI in particular), the grey-white matter contrast isn’t very strong, so:

* In the Actions panel, select Clustering and create a background/foreground cluster
* For Number of Clusters, select 2 (default is 3), leave “Foreground cluster” 1.
* Click on the blue-background image in the viewer. Blue means empty (this should be everything but the brain). Click on the icon above the camera to view the images side-by-side.
* Click Next
* You can add a few bubbles for it to grow from. Change “Step size” to 10 to make it faster and press play. When you are happy pause it and click Finish.
* On the top menu, click Segmentation and then Save Segmentation Image…

This procedure will automatically reject the edge slices (top and bottom), and when the intensity in the middle slices is flat it will try to cluster them together.

You might miss some regions that have very low intensities, like the veins, but you will fill these in the next step.

Once you have created a custom mask using the procedure above, run the following ANTs commands in the command line, which will dilate/erode the mask:

* **ImageMath 3 custom\_mask.nii.gz MC custom\_mask.nii.gz 4**

MC = Morphological Closing -> it does dilation and erosion iteratively; 4 = radius

* **ImageMath 3 custom\_mask.nii.gz FillHoles custom\_mask.nii.gz** (fills holes in the mask)
* **ImageMath 3 custom\_mask.nii.gz MD custom\_mask.nii.gz 1**

MD = Morphological Dilation -> This makes sure you have a little bit more brain mask

# Motion and distortion correction

**sk\_ants\_Realign\_Estimate.sh** estimates and corrects for inter-volume motion for both EPI and opposite phase-encoding images. First, a motion corrected mean template is created for both functional and opposite phase-encoding images (\**MoCorr\_meanTemplate.nii.gz*).

The opposite phase-encoding mean template is then registered (rigidly) to the functional mean template resulting in a \**MoCorr\_meanTemplate\_reg2func.nii.gz* file. This is important as the subject will have moved between the start of the functional data and the opposite-phase data.

Both the functional mean and the \**reg2func.nii.gz* templates are used for the distortion correction, which is carried out using the ANTs script **antsMultivariateTemplateConstruction2** and performs a ‘meet-in-the-middle' registration using ANTs’s SyN algorithm.

The resulting transformations named \**00GenericAffine.mat* and \*1Warp.nii.gz encode the affine transformation and distortion warp, respectively.

In addition, this step will also generate a \**DistCorr\_template0.nii.gz* file which consists of a motion-corrected and distortion-corrected functional template image.

This script will also save the transformation matrices for each and every volume in a folder called \*\_mats. These matrices encode rigid/affine motion estimates.

An additional folder called \*\_split will also be created. This folder will include the individual 3D volumes. This is done to prevent processing from being limited by the computer’s RAM.

Additional runs of a specific session will be similarly aligned with sk\_ants\_Realign\_Estimate.sh, but using the average functional and mask of the reference run of that session. This will implicitly align every single run to a specific reference run of each session. For example, if run1 is the reference run of day1, then all additional runs acquired on day1 will be implicitly aligned with run1 of day1. Likewise, if run3 of day2 is the reference run for that day, all additional runs acquired on day2 will be implicitly aligned with run3 of day2.

Here is a short explanation of the options in **sk\_Estimate\_Realign.sh**:

# MP2RAGE preprocessing

The mp2rage image will be cleaned up (i.e., noisy background removed).

The wrapper MPRAGEise.m will run the presurfer scripts that use SPM functions to clean up the background, extract WM tissue (important for the BBR step, as explained below) and creates the appropriate anatomical brain mask.

The two important files from this step are: \**MP2RAGEised.nii* and \**MPRAGEised\_WMmask.nii* (which will subsequently be compressed).

# Functional-to-anatomical alignment

By default, the script will perform functional-to-anatomical registration using FSL’s bbr (boundary-based registration) algorithm included in flirt, although you will have the option to choose a manual registration file and align with **sk\_antsFineReg.sh** in case bbr fails (see below how to do the manual registration).

All the transformations will then be converted to ITK-like, so that they can be passed to the last step of the pipeline which is applying the transformations and reslicing the original unprocessed functional data.

## BBR vs FineReg

When it comes to transforming the functional to anatomical, it is recommended to use the BBR method, especially if you’re interested in the ventral and medial temporal lobes. This is because the presence of signal dropouts can cause the mutual information based registration to be sub-optimal. BBR cost-function uses additional information in the form of the WM mask and uses the signals sampled at the boundaries to optimise the registration between EPI-anatomical images. In our experience, BBR more consistently provides good alignment for medial-ventral temporal lobe datasets.

However, always make sure you check the output of BBR once the func2anat registration files are written out. This can be achieved by simply loading the resulting \**fslBBR\_Warped.nii.gz* overlaid on the mpragised image:

**itksnap -g sub001\_UNI\_MPRAGEised.nii.gz -o sub001\_run1\_AP\_DistCorr\_template0\_fslBBR\_Warped.nii.gz**

Regardless of whether you use BBR or FineReg, the fixed image is always the mpragised anatomical image, and the moving image will be the distorted-and-motion-corrected template image (i.e., \**DistCor\_template0.nii.gz*) computed using the **sk\_ants\_Realign\_Estimate.sh** script.

FSL’s flirt function will be used to calculate a rigid transformation from functional to anatomical space using the BBR cost function. This will be saved as \**DistCorr\_template0\_fslBBR-to-ITK.mat* after converting the FSL matrix to an ITK-compatible matrix file. This is necessary because we will use ANTs to apply all transformations in a single step.

Functional-to-anatomy registration will only be done for the study reference run (in the ref\_dayrun dictionary this will be the value you associate with the key ‘ref’).

# Manual Registration

After the **sk\_ants\_Realign\_Estimate.sh** script finishes the estimation, you will find a file named \**DistCorr\_template0.nii.gz* in your working directory. This is a template file that has been motion-corrected and with distortions corrected-for. You should check that the estimation and alignment steps were successful by doing the following steps:

* After cleaning up the MP2RAGE a mprageised version will be saved to the working directory. Open it in itksnap and then choose the \**DistCorr\_template0.nii.gz* as an overlay

**itksnap -g sub001\_uni\_MPRAGEised.nii.gz -o sub001\_run1\_AP\_DistCorr\_template0.nii.gz**

* You can also optionally change colours just to spot it more easily:
  + Tools – Layer Inspector
  + Click on down-arrow – Color map – choose colour set
* Check if the anatomical and EPI are fairly well aligned
* Click Ctrl-R (Tools - Registration) to open the registration tab

If more or less aligned (meaning, the functional slab and anatomical data show overlap and good correspondence of anatomical features), run the manual registration step in ITKSNAP:

* First, check if Automatic does a good job
  + Transformation should be Rigid and similarity metric “Mutual information”
  + Choose 1x Coarsest level
  + Click on Run Registration
  + Check result, if not good, change coarse level to 2x (lower resolution) and then we choose a finer resolution
* Click on the Manual tab
* Move the brain around a bit and try to align some visible anatomical features , for example, the ventricles or whatever brain region is most relevant to you (e.g., hippocampus):
  + You can also rotate using the circle axis
  + Click on “Reset to identity” to go back to initial
  + When you are happy with the starting point, save it (e.g., init.txt) clicking on the floppy disk / save icon on the Manual tab
  + Then click on Automatic tab and Run Registration – it will use the values you set above

# Inter-run registrationi

For inter-run registration (e.g., aligning functional runs from day2 to day1, or from the same day) the script aligns the reference functional run of one session to a reference functional run of the same or another session (e.g., first run on the first day; this is defined by you at the beginning of the master python script).

This saves the inter-run registration mat (*\*antsFineReg\_0GenericAffine.mat*) and the inter-run warp image (*\*antsFineReg\_Warped.nii.gz*).

Inter-run registration is done using the script **sk\_antsFineReg.sh:**

* -f: this is the template of reference run (e.g., sub01\_run1\_AP\_DistCorr\_template0.nii.gz)
* -g: mask of the reference functional run (e.g., sub01\_run1\_AP \_avg\_n4\_mask.nii.gz)
* -m: template of another functional run that you would like to register to the reference run (e.g., sub01\_run4\_AP\_DistCorr\_template0.nii.gz)
* -n: mask of the another functional run (e.g., sub01\_run4\_AP \_avg\_n4\_mask.nii.gz)
* -i: initial manual registration matrix (e.g., from itksnap)
* -x: should the registration be done automatically (1) or not (0)?

If you choose 1, you should also include the masks (-g and -n options).

If you choose 0, you need to include an initial manual registration matrix (-i option)

Case 1: Aligning runs from the same day as study reference run

Because they were collected on the same day, they will all be on the same space and will have the same orientation, so the script will simply run **sk\_ants\_Estimate\_Realign.sh using the study reference** . For example, if run1 (study reference run) and run2 were acquired on the same day:

**sk\_ants\_Realign\_Estimate.sh -f day1run1\_AP\_avg\_n4.nii.gz -x day1run1\_AP\_avg\_n4\_mask.nii.gz -a day1run2\_AP.nii.gz -b day1run2\_PA.nii.gz**

This will implicitly align create a mat file run2 to run1. Although it’s unlikely to fail, you should still check the registration visually!Check the result with:

itksnap -g run1\_AP\_DistCor\_template0.nii.gz -o run2\_AP\_DistCor\_template0.nii.gz

If you notice that the alignment is sub-optimal, you’ll have to do a manual-guided alignment by following the “Manual registration” section above, and save the registration matrix.

Case 2: Aligning runs from the different days

For inter-run registration, the process is slightly more involved. First, the script **sk\_ants\_FineReg.sh** will be used to compute a transformation matrix from a reference functional run from a specific day (e.g., run1 of day2) to the study reference run from another day (e.g., run1 of day1).

Since all additional runs of day2 have been implicitly aligned with the reference run of day2 (during the **sk\_ants\_Estimate\_Realign.sh**), you only need to run this step once, because you just need to get a starting point from day2 to day1.

The fine registration step will then be automated using **sk\_antsFineReg.sh** script. This is following the assumption that the amount of translation/rotation between runs of different days is much larger than runs collected within the same day.

The master python script uses the automatic alignment by default (i.e., -x 1) to align functional runs of different days. However, if this fails, you should perform a manual registration using itksnap as explained above. You have the option to input a manual registration file in the master script.

After you generate the day2-to-day1 initialisation matrix, *manual\_init.mat* file from the manual alignment (e.g., using itksnap) and define it in the master script, the pipeline will use this initialization matrix in **sk\_ants\_FineReg.sh** (using the -i flag); ANTs will initially move the functional run of day2 to the reference run of day1 according to the values in *manual\_init.mat* and then will compute an automated registration:

**sk\_ants\_FineReg -f day1run1\_AP\_DistCorr\_template0.nii.gz -m day2run1\_AP\_DistCorr\_template0.nii.gz -i manual\_init.mat**

After **sk\_ants\_FineReg.sh** is completed, it will spit out a file named \**antsFineReg\_0GenericAffine.mat*. This is the rigid alignment of, for example, a functional run on day2 to a reference functional run on day 1.

# Reslicing

The transformations computed throughout the script will only be applied at this stage (note that no reslicing happens in any stage in the processing, other than now).

Reslicing is done using the script **sk\_ants\_Realign\_Reslice.sh**, which options are:

-r: This is the template from the reference run (e.g., day1\_run1\_template0.nii.gz)

-u: This is func-to-func\_ref mat file (day2\_run1\_antsReg\_0GenericAffine.mat).

For example, if you have a registration from a functional run on day 2 (func), to a functional run on day 1 (func\_ref), you should add the registration file here

-f: this is the anatomical mprageised image (e.g., anat\_mpragised.nii.gz)

-x: This is the func-to-anatomy registration file (e.g., day2\_run1\_bbr2itk.mat)

-a: this is the original unprocessed functional data

After running **sk\_Realign\_Reslice.sh**, a mean of the final anatomically-aligned functional time series is computed for every single run, so that you can compare it to the anatomical image.

# Output files

After the entire script has completed, a json file named *VARS.json* will be written to the working directory. This file will show you what files are most important:

**func**: original functional unprocessed data (e.g., /path/to/data /run3\_AP.nii.gz, or, if trimmed, ., /path/to/data /run3\_AP\_20vols.nii.gz)

**opp**: original opposite PE unprocessed data (e.g., /workdir/run3\_PA.nii.gz)

**avg\_func**: average of functional raw data (e.g., /workdir/run3\_AP\_avg.nii.gz)

**avg\_opp**: average of functional opposite PE data (e.g., /workdir/run3\_PA\_avg.nii.gz)

**n4\_avg\_func**: average functional raw data with bias field removed (e.g., /workdir/run3\_AP\_avg\_n4.nii.gz)

**mask\_avg\_func**: mask of average functional raw image (e.g., /workdir/run3\_AP\_avg\_n4\_mask.nii.gz)

**n4\_avg\_opp**: average opposite PE raw data with bias field removed (e.g., /workdir/run3\_PA\_avg\_n4.nii.gz)

**mask\_avg\_opp**: mask of average functional raw image (e.g., /workdir/run3\_PA\_avg\_n4\_mask.nii.gz)

**distCor\_mat**: transformation matrix encoding motion and distortion correction (e.g., /workdir/run3\_AP\_DistCorr\_00GenericAffine.mat)

**temp0**: motion- and distortion-corrected image template (e.g., /workdir/run3\_AP\_DistCorr\_template0.nii.gz)

**interrun\_init**: transformation matrix from a functional run to the reference functional image (e.g., /workdir/run3-to-run1\_fslBBR\_init.mat)

**interun\_mat**: transformation matrix from a functional image of one run to a reference functional image (e.g., /workdir/run3\_AP\_DistCorr\_template0\_antsFineReg\_0GenericAffine.mat)

**interrun\_warp**: functional template image aligned to a reference functional image (e.g., /workdir/run3\_AP\_DistCorr\_template0\_antsFineReg\_Warped.nii.gz)

**bbr\_mat**: transformation matrix from functional to anatomical space (e.g., /workdir/run3\_AP\_DistCorr\_template0\_fslBBR.mat)

**bbr\_warp**: functional template image moved to anatomical space (e.g., /workdir/run3\_AP\_DistCorr\_template0\_fslBBR\_Warped.nii.gz)

**bbr\_itk**: transformation matrix from functional to anatomical space converted to itk (e.g., /workdir/run3\_AP\_DistCorr\_template0\_fslBBR-to-ITK.mat)

**func\_anat\_space**: functional data resliced and moved to anatomical space (e.g., /workdir/run3\_AP\_bbrreg\_MotDistCor\_anatAligned.nii.gz)

**func\_epi\_space**: functional data resliced and moved to functional space (e.g., /workdir/run3\_AP\_bbrreg\_MotDistCor\_funcAligned.nii.gz)

In addition, a text file called *CMD.txt* will also be saved to the working directory, which will detail every single step of the pipeline, so that you can reproduce the results from the command line.

The first few lines might look like this:

# Copying /path/to/data/run1\_AP.nii.gz into /workdir/run1\_AP.nii.gz

# Copying /path/to/data/run1\_PA.nii.gz into /workdir/run1\_PA.nii.gz

fslroi /workdir/run1\_AP.nii.gz /workdir/run1\_AP\_20vols.nii.gz 0 20

antsMotionCorr -d 3 -a /workdir/run1\_AP\_20vols.nii.gz -o /workdir/run1\_AP\_20vols\_avg.nii.gz

antsMotionCorr -d 3 -a /workdir/run11\_PA.nii.gz -o /workdir/run1\_PA\_avg.nii.gz

sk\_antsBET.sh -i /workdir/run1\_AP\_20vols\_avg.nii.gz

etc.