Manual for spatial transformation procedure and quality assessment of spatial transformations of functional and structural locus coeruleus (LC) magnetic resonance imaging (MRI) data

Main text:
It is the Locus Coeruleus! Or… is it?: A proposition for analyses and reporting standards for structural and functional magnetic resonance imaging of the noradrenergic Locus Coeruleus
Yeo-Jin Yi, Falk Lüsebrink, Mareike Ludwig, Anne Maaß, Gabriel Ziegler, Renat Yakupov, Michael Kreißl, Matthew Betts, Oliver Speck, Emrah Düzel*, Dorothea Hämmerer* (*shared last)

1. Introduction

1.1. Background and aim of this procedure

The LC is so small that even a slight spatial deviation can significantly reduce the statistical power of group-level analyses. For more robust functional assessments, it is not only important to register and normalise each single subject functional image precisely, but also to control the quality of the transformed images and rectify any problems that reduce the precision of the spatial transformation. With the transformation and quality assessment protocol introduced in detail below, we show a way how to achieve more precise spatial transformations for functional and structural LC imaging data (with deviations on average below 1mm), which will allow for more reliable assessments of functional LC imaging data at the group level.

1.2. Spatial transformation procedure

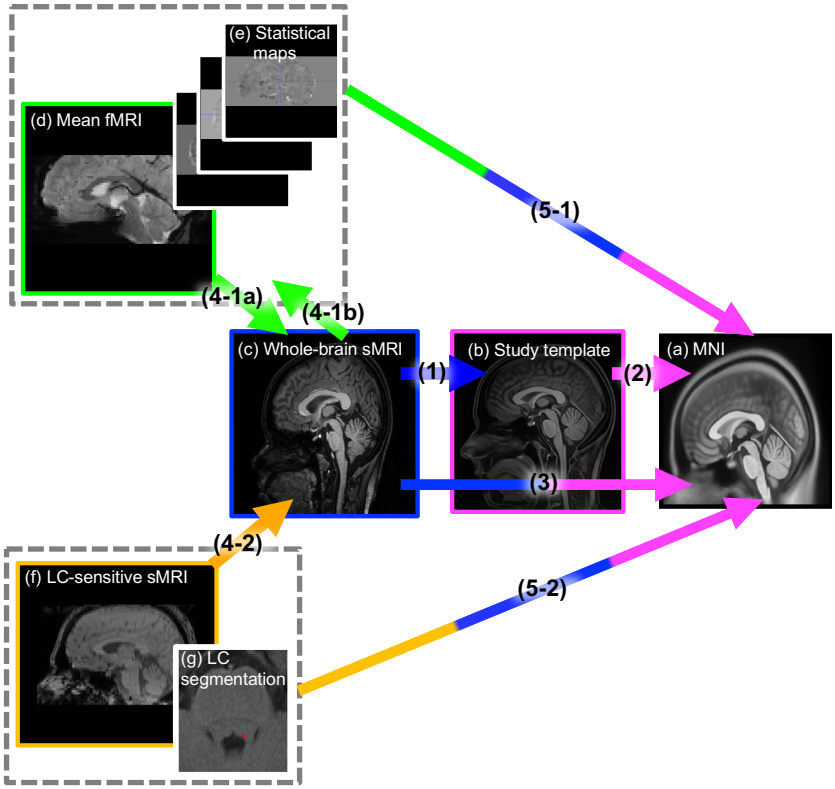


Figure 1. A schematic of the spatial transformation procedure

First, pre-processing of the images to prepare for the spatial transformation is required. Then, a series of transformations are executed on the mean functional images (Figure 1d), whole-brain structural images (Figure 1c), LC-sensitive structural images (Figure 1f), and LC segmentations (Figure 1g). Scripts for these transformation steps are downloadable via this link: https://github.com/alex-yi-writes/LC-SpatialTransformation2021

Functional images are slice-time corrected, unwarped with distortion fields calculated from the double-echo gradient echo field map and realigned to the mean volume with Statistical Parametric Mapping (SPM12, http://www.fil.ion.ucl.ac.uk/spm12.html) in the MATLAB environment (Version 2015a, Mathworks, Sherborn, MA, USA, 2015) using default parameters. This generates a mean functional image in native voxel size per person and per scan session, used for spatial transformation of the structural and functional images (cf. Figure 1d). Thereafter, the time-series functional images are smoothed with SPM using a kernel that is slightly bigger than the voxel size (e.g. a 3×3×3mm kernel for 2mm^3 voxel resolution or 2×2×2mm for 1.5mm^3 voxel resolution), before running single-subject general linear models (GLM) to estimate task-related contrasts in SPM. In running the GLM, artefacts due to physiological noise and movement should be corrected by generating physiological noise regressors from physiological recordings acquired during the scan or utilising a component-based analysis method (e.g. CompCor [Behzadi et al., 2007] or FSL MELODIC [Beckmann & Smith, 2004]). The GLM analysis generates a set of statistical contrast maps in the native space per subject (Figure 1e) that is ready for transformation into the MNI space.

Individual T1-weighted whole-brain structural images are bias-corrected using Advanced Normalisation Tools' N4BiasFieldCorrection function (Figure 1c) to correct for RF-field-related inhomogeneity (ANTs, Version 2.3.1; http://picsl.upenn.edu/software/ants/, 2016; Avants et al., 2011; Tustison et al., 2010). A study-specific template (Figure 1b) is created from these bias field-corrected structural whole-brain images using the *antsMultivariateTemplateConstruction2.sh* function of ANTs (Avants et al., 2011).

The LC is manually segmented on the individual neuromelanin-sensitive images (cf. Figure 1f and 1g) using ITK-Snap software (version 3.6.0-RC1; http://www.itksnap.org, 2018) by two or more independent expert raters. Final LC segmentations may contain only the overlapping voxels from all raters (cf. Figure 1g) (see also Hämmerer et al. [2018] for more details on LC segmentation generation).

In contrast to the functional LC data, structural LC imaging data might be acquired at anisotropic resolutions with voxels oriented parallel to the length of the LC to increase SNR (Sasaki et al., 2006). To facilitate spatial transformations across scans that were acquired at different resolutions, LC-sensitive structural images and LC segmentations should be re-sampled to a voxel size matching the structural MNI template or a study-specific template

Commented [DH1]: In the manual, one should write in present, please alter accordingly

resolution using the mri_convert function in FreeSurfer (Version 7.1; http://surfer.nmr.mgh.harvard.edu/, Martinos Center for Biomedical Imaging, Charlestown, Massachusetts). Finally, brain-only binary masks are created from the mean functional images using the bet2 function in FSL (version 6.0.1; https://fsl.fmrib.ox.ac.uk/fsl/, Analysis Group, FMRIB, Oxford, UK) to aid normalisation by increasing the geometric compatibility among the images and limit the image areas that are considered and transformed by the similarity metric.

Having prepared all relevant data, the following procedure will allow for an adequately precise spatial transformation of structural and functional LC data to MNI space (or a study-specific template space, if so desired). First, the individual whole-brain structural MPRAGE image (Figure 1c) in the native space is registered non-linearly to the study-specific template (Figure 1b) using *antsRegistrationSyN.sh* (step 1 in Figure 1). To prepare the transition from the native to the MNI space, the study-specific template (Figure 1b) is non-linearly registered to the MNI space using *antsRegistrationSyN.sh* (step 2) and individual whole-brain structural images (Figure 1c) are transformed into the MNI space (Figure 1a) with the concatenated transformation matrix and deformation fields generated from steps 1 and 2 using *antsApplyTransforms* (step 3). Then, in the case of whole-brain functional images, the individual mean functional image (Figure 1d) is rigidly registered to the individual whole-brain structural image (Figure 1c) (step 4-1a). The direction of registration in this step is reversed in the case of partial volume functional images with a narrow field of view (step 4-1b). The conventional method of registering two images together is to register a lower-resolution image to a higher resolution image. However, as there is a reduced degree of common features between a partial-volume image and a whole-brain image, registering the former to the latter is more prone to errors compared to the same procedure with two whole-brain images (Avants et al., 2011; Tustison & Avants, 2013). With the concatenated transformation matrices and deformation fields acquired from steps 1, 2, and 4-1a/b, the mean functional images (Figure 1d) and individual contrast images (Figure 1e), which are in the same space as individual mean functional images, are non-linearly transformed to the MNI space (Figure 1a) in one transformation step using *antsApplyTransforms* (step 5-1). Group-level voxel-wise analyses for LC activations can then be performed on these contrast images that have been moved into the MNI space.

Similarly, the LC-sensitive structural image (Figure 1f), and the LC segmentation (Figure 1g) in the space of the LC-sensitive structural image are rigidly registered to the individual structural whole-brain image (Figure 1c) using antsRegistrationSyN.sh (step 4-2). Then, concatenated transformation matrices and deformation fields acquired from steps 1, 2, and 4-2 are used to transform LC segmentations (Figure 1g) delineated on LC-sensitive structural images (Figure 1f) non-linearly into the MNI space (step 5-2) using *antsApplyTransforms*. All

nonlinear spatial transformations are implemented at the 4th degree B-spline interpolation except the individual contrast images (Figure 1e), which are transformed with the linear interpolation option. Individual LC segmentations (Figure 1g) are transformed with the nearest neighbour option to avoid creating voxels in interpolation steps with unclear association with the original LC segmentation. Note that by following a similar approach but omitting the transformation step to MNI space in the concatenated transformation matrices, group analyses for structural and functional data can also be done with comparable precision in study-specific template space (Figure 1b) (Suppl. Figure 3 of the main text article). For further details regarding the transformation parameters and how to adjust them in case of altered contrast properties of the used imaging sequences, see the code downloadable at https://github.com/alex-yi-writes/LC-SpatialTransformation2021 .

1.3. Overview of landmark assessment procedure

After all subject datasets have been processed, the quality of the transformation must be examined. This examination is done by selecting the landmarks on the structural MNI space template, drawing them on the individual mean functional images transformed into MNI space, and calculating Euclidean distances between the template-defined landmarks and their single-subject counterparts. Additionally, prior to the landmark assessment, a video that takes single-subject transformed functional or structural images per frame can be made and inspected quickly to identify abnormal registrations in individual images. More detailed procedures will be explained in the walkthrough section.

2. Transformation procedure

2.1. Preparation

2.1.1. Required data

The files that are needed for the assessment are:

- the structural MNI template in which space all images are transformed
- transformed mean functional images of each subject

The structural MNI template can also be a study-specific template (i.e. study template (b) in Figure 1), depending on the analysis that will be done with the transformed functional data. In this example analysis, a structural MNI template created by Fonov et al (2011) was used.

2.1.2. Required software

To draw the landmarks, we will use:

- ITK-SNAP (version later than 3.6.0-RC1; http://www.itksnap.org), or any segmentation-enabled brain image viewers
- MATLAB (version later than 2013b, Mathworks, Sherborn, MA, USA), or any computational programmes that can read Nifti images

The example MATLAB codes for generating the overlay video and statistical assessment is in section 3.

2.2. Quality assessment Procedure

2.2.1. Selecting landmarks

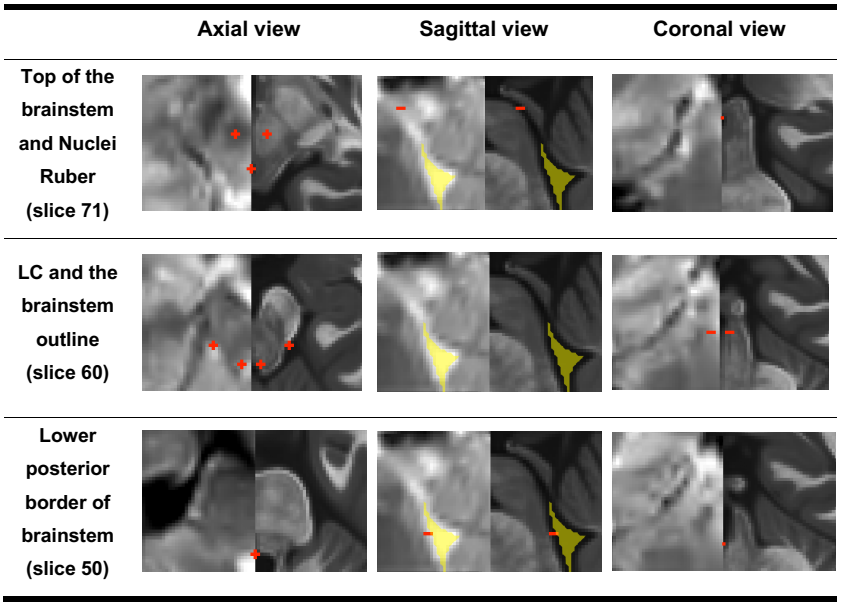
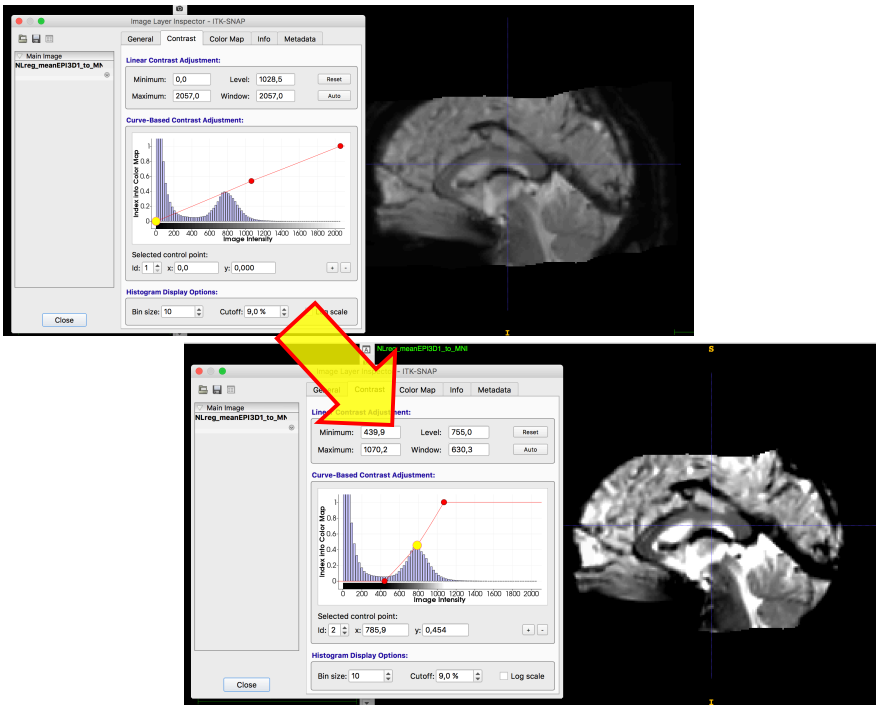


Figure 2. Selecting landmarks for functional data

Selecting an appropriate set of landmarks depends on the region of interest (ROI). This manual's ROI, the LC, is in the brainstem near the lateral floor of the fourth ventricle. Therefore, selecting landmarks in the upper brainstem that delineate the bounds of LC location is most helpful for the evaluation process. In addition, the landmarks should be clearly visible and anatomically distinguishable compared to the surrounding structures on the mean *functional* images, as these landmarks are drawn manually through visual inspection on the functional images. Based on these criteria, the landmarks specified in the table above were selected and drawn first on the structural MNI template.

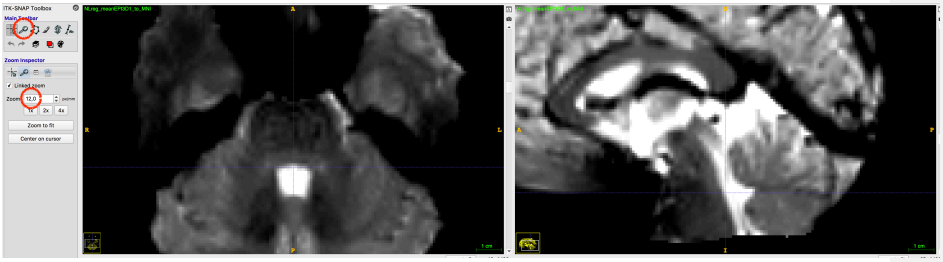
The top of the brainstem and the nuclei ruber were selected as a landmark to indicate the upper bound of the larger ROI area of the study, the brainstem. Nucleus ruber is distinctly visible in the functional images, which makes it a good candidate for a landmark. The outline of the brainstem can be placed anywhere that are distinctly visible along the border of the brainstem. Once the landmarks are selected, evaluators of the landmarks should discuss the

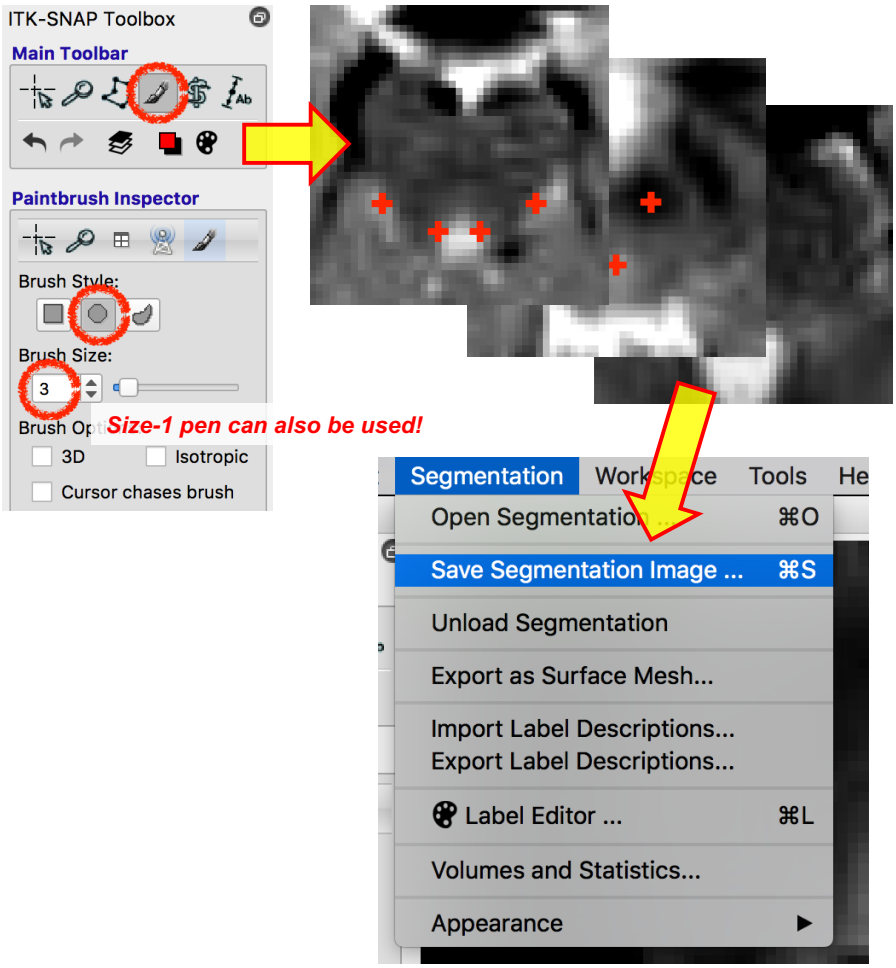
rating strategies of landmarks in the MNI space and sufficiently familiarise themselves with them before placing the landmarks in the transformed mean functional images.



2.2.2. Drawing landmarks

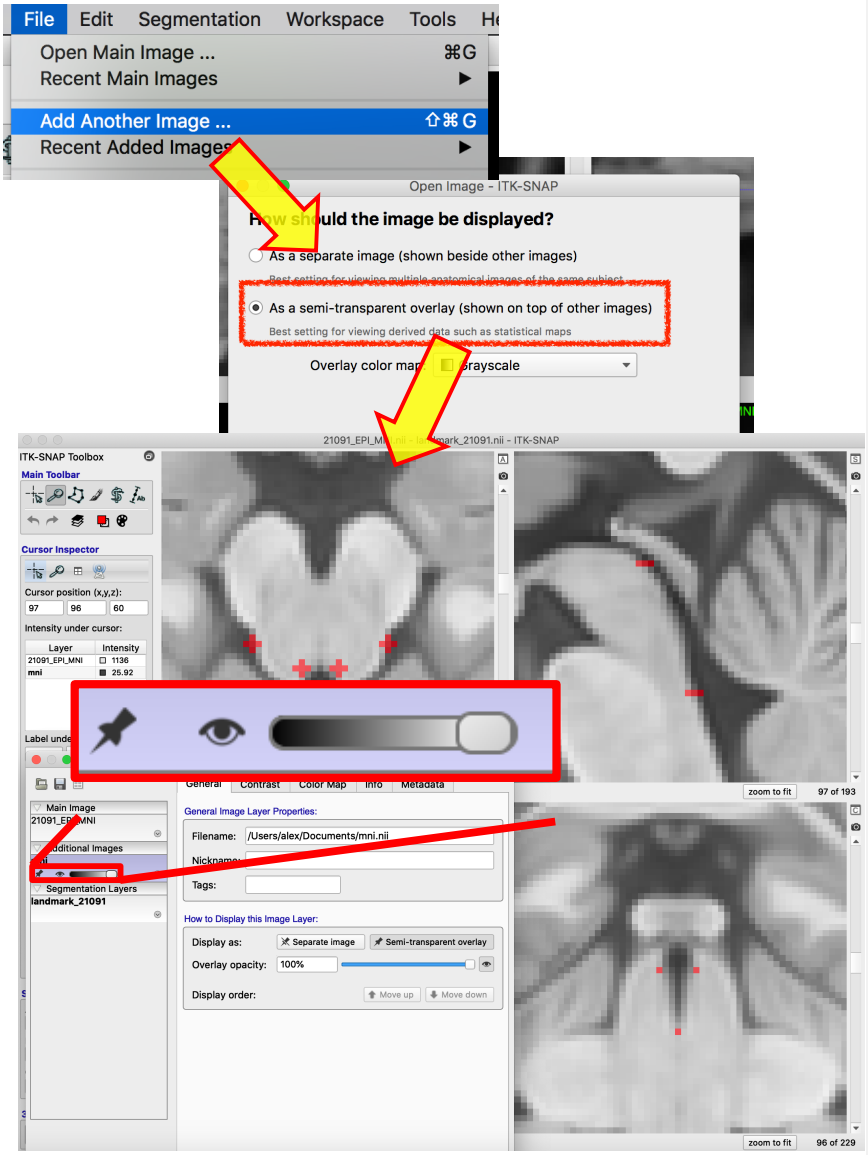
Next, evaluators inspect one of the transformed single-subject mean functional images (c.f. Figure 1, **(d)** in **(a)** space). If necessary, adjust the contrast slider so that the fourth ventricle is as bright and distinguishable as it can be. After adjusting the contrast of the image, magnify the image so that you can draw the landmarks (Figure 2) more precisely.





Then, using the size-3 or size-1 round brush segmentation tool, place the landmarks on the transformed mean functional images in axial view, on the pre-defined slices of each landmark. Use the anatomical clues from other views to aid the landmark placement. For example, when drawing the lower posterior border of the brainstem, the deepest groove of the fourth ventricle can be a guiding structure. When the assessment of the single-subject image is complete, save the landmarks as a segmentation image.

Optionally, after saving the image, evaluators could open the group space image used for transforming the mean functional images as a semi-transparent overlay. Once all images are transformed, take note of the subject identifiers if the landmarks were excessively out of



place. However, **DO NOT** change the landmark image you already saved, as it indicates the quality of the functional image transformation.

2.2.3. Aggregating single-subject landmarks for preliminary inspection

When the assessment of the whole dataset is complete, visual inspection of all landmarks aggregated in a same space is performed by calculating a heatmap of the average of the saved individual landmarks (c.f. Figure 3, the MATLAB code from section 4 of this manual). Additionally, a frame-by-frame video of the transformed images (the code to generate the video is in section 3.3 of this manual, and an example of this video is in the section 2.4 of the supplementary material) can be generated, which allows for identifying outliers easily.

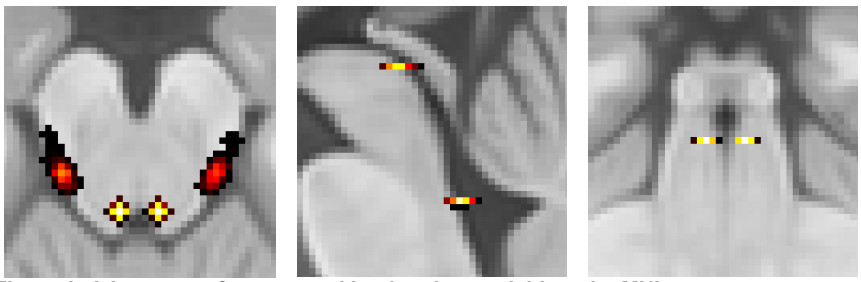


Figure 3. A heatmap of aggregated landmarks overlaid on the MNI space

2.2.4. Statistical quality assessment of aggregated landmarks of transformed functional images

Once all faulty transformation issues are addressed and rectified (see section 2.3 of this manual for how to correct potential transformation failures individually), statistical analyses of the quality assessment are performed to quantify the transformation precision. This is done by calculating each of the in-plane distance between individual landmarks on the mean functional images and the landmarks set on the structural MNI or group template. An example code for the analyses is in the section 3.4. Through this step, the spatial transformation can be further quantified. The detailed background and results for this example analysis are described in the main text and in Figure 4.

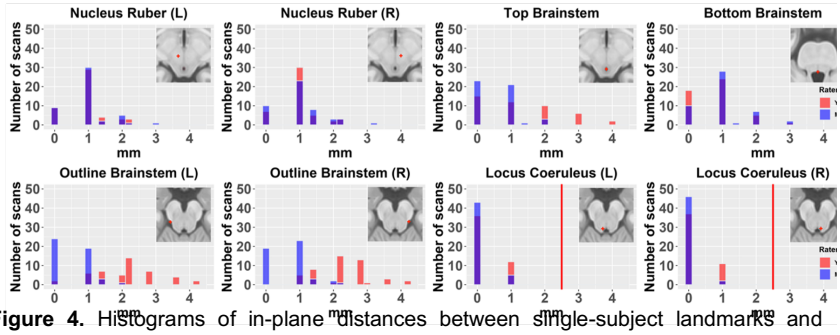


Figure 4. Histograms of in-plane distances between single-subject landmarks and landmarks defined on the MNI template in an example dataset. The median of in-plane distances should be at 2mm or lower for all landmarks, thereby falling below the typical width of the LC of 2.5mm (in the LC-focused landmarks, indicated by the solid red line, Fernandes et al., 2012). Note that deviations in the outline of the brainstem are bound to differ more from the MNI marks as the precise position along the border of the brainstem is less relevant than capturing the border between brainstem and CSF (cf. row 2 in A). See section 2.2.2 of this manual for details on how to set and evaluate landmarks.

2.2.5. Statistical quality assessment of transformed structural images

Similarly, the spatial transformation quality of the structural images can be assessed using the transformed LC segmentations. As specified in the section 2.5 of the main text and the section 1.2 of this manual, the LC segmentations are transformed into MNI space using the transformation matrices and the deformation fields generated from warping neuromelanin-sensitive structural images (Figure 1f) to the MNI space (step 5-2). The quantification of the spatial transformation precision can be done by calculating in-plane distances between the slice-wise centroids of the template LC mask in the MNI space, e.g., the meta LC template by Dahl et al. (2022), was done here, and the transformed LC segmentations of each subject. An example code for calculating the distance is in the section 3.5 (see also figure 5 for an example of results of this analysis).

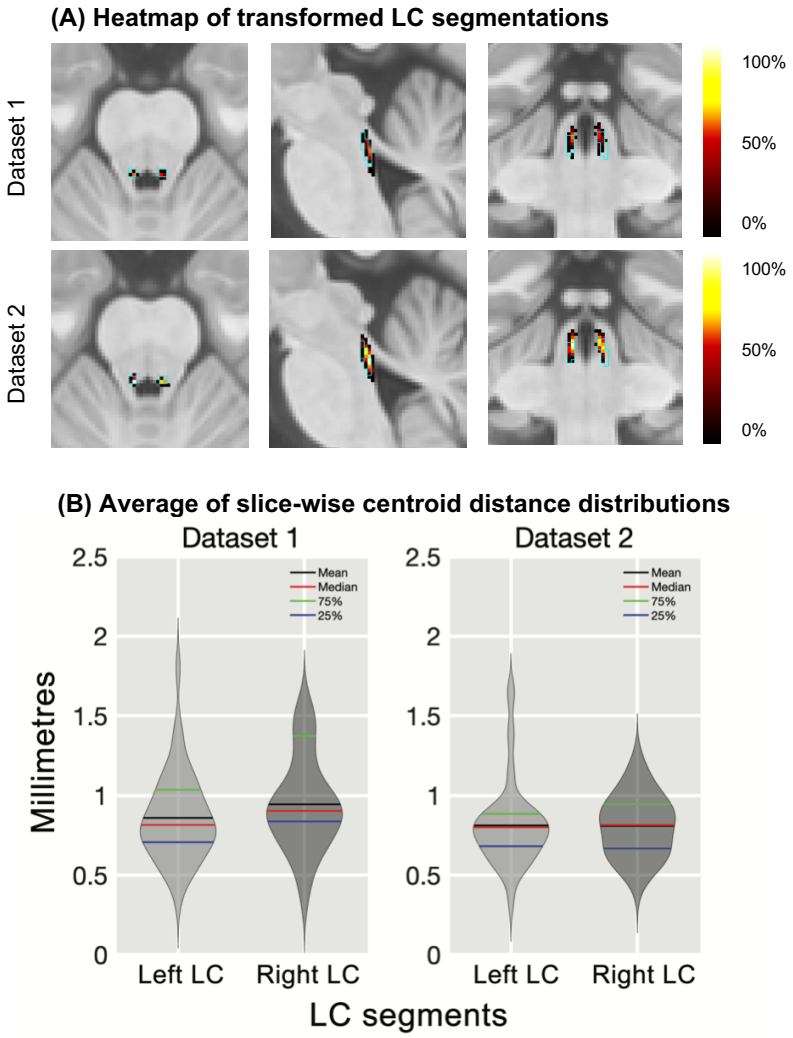


Figure 4. (**A**) A heatmap of transformed individual LC segmentations in the group space. The cyan line indicates the meta LC mask created by Dahl and colleagues (2022). The maximum overlap across transformed segmentations is at 54.1% and the minimum at 2% in dataset 1, and maximum 62% and minimum 2% in dataset 2. This low overlap can be attributed to the difference in each rater's conservativeness in segmentation evaluation criteria (mean included voxel: [dataset 1: YY=12.77; DH=11.83], [dataset 2: AH=43.34; JK=48.64]). (**B**) Violin plots showing the distribution of distances across subjects for the left and right LC centroid voxels of aggregated meta LC mask and MNI-transformed single-subject LC segmentations in each dataset. The in-plane distance is calculated slice-by-slice separately for left and right LC and averaged across slices to yield one value per subject and left or right LC segmentation ([dataset 1: left: *M*±*SD*=0.85±0.27, *IQR*=0.29; right:

M±*SD*=0.94±0.28, *IQR*=0.34], [dataset 2: left: *M*±*SD*=0.81±0.27, *IQR*=0.21; right: *M*±*SD*=0.81±0.19, *IQR*=0.28]). A plot showing all cases of slice-wise distances across all LC segmentations and slices can be found in Supplementary Figure 2.

2.3. Examples of spatial transformation failures and fixes

The spatial transformation pipeline included in this manual already contains extended measures that troubleshoot possible errors. In this section, examples of errors are described, in case they arise in a pipeline that is tailored to a large dataset and pruned for efficiency.

2.3.1. Suboptimal registration of mean functional images

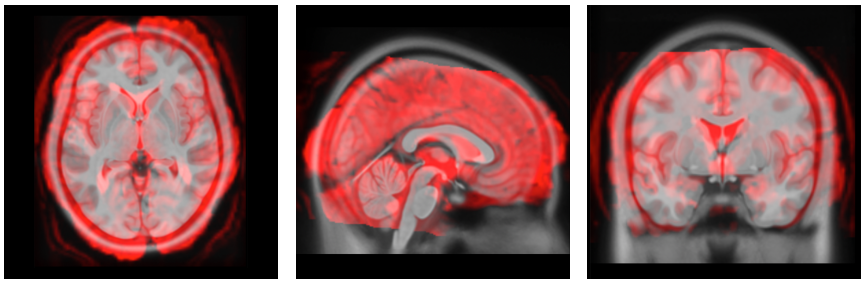


Figure 4. An example of suboptimal registration in the transformed mean functional image. The transformed mean functional image is indicated as red additive overlays on the MNI template image in each view.

One example of errors that could occur is that the brain of the transformed mean functional images appears to extend over the bounds of the tissue area of the MNI template. When backtracking the transformation steps to isolate the error source, the failure can be seen during the step 4-1 of Figure 1, rigid registration of the structural whole-brain image to the mean functional image. From the images below, it can be seen the skull of the structural image is fitted to the tissue of the mean functional image.

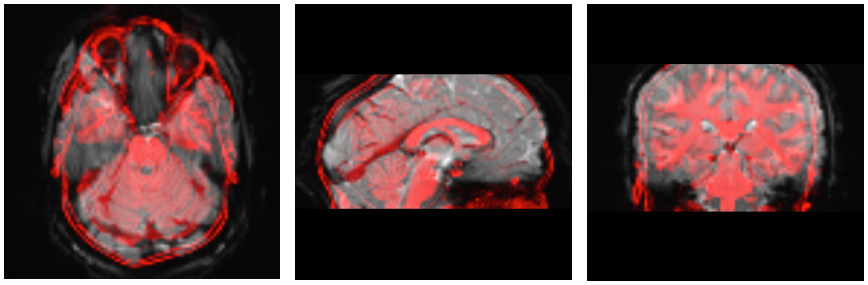


Figure 5. Rigid registration failure of the structural whole-brain image to the mean functional image. The structural image was registered to the native mean functional image and is indicated as red additive overlays on the mean functional image.

This issue seems to occur because the *antsRegistrationSyN* algorhithm used for this step perceives the intensity of the tissues near the skull in the mean functional image to be more similar to the skull of the structural image than the faint traces of the skull in the mean functional image. The problem can be solved by including the brain-only mask of the mean functional image to the *antsRegistrationSyN* call (line 42 of the pipeline) or skull-stripping the images involved in the registration process.

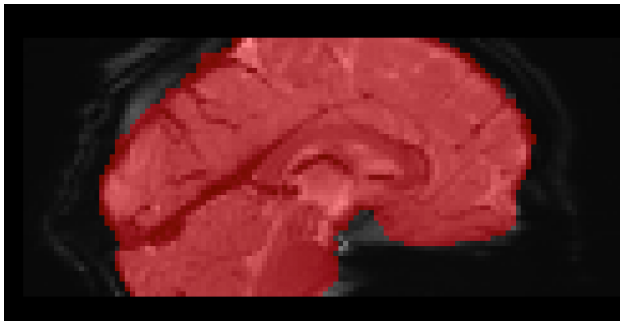


Figure 6. An example brain-only mean functional image mask overlaid on the mean functional image. The transparent red image indicates the area where the mask covers.

There are various methods to generate the brain-only mask, and any tool can be chosen for this measure as far as the mask properly includes all viable tissues in the image. Once the mask is generated, the step 4-1 and 5-1 (c.f. Figure 1) are executed with the new files, and the quality assessment is done once again on the new final output. The below is the example of the transformed mean functional image processed with the brain-only functional image mask.

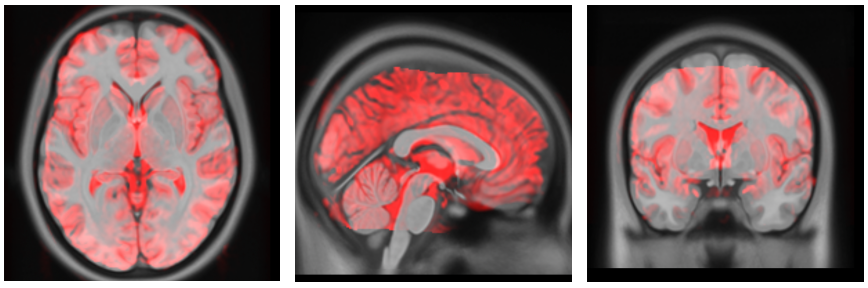


Figure 7. An example of successful spatial transformation of the mean functional image after employing the brain-only mask in the step 4-1 of Figure 1. The transformed mean functional image is indicated as red additive overlays on the MNI template in each view.

2.3.2. Suboptimal normalisation near the skull in mean functional images

Another example of transformation failure is faulty normalisation of the cortical surface area near the skull in the transformed mean functional images. The cause of this problem is similar to that of the issues described in the section 2.3.1.

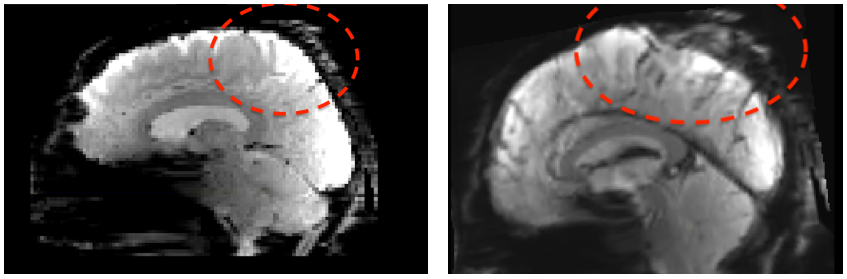


Figure 8. An example of suboptimal normalisation in the mean functional images. The left pane shows the native mean functional image, and the right pane shows the transformed mean functional image.

The same solution, applying the brain-only mask to guide the mean functional image normalisation can be employed. However, if the problem persists, skull-stripping all images, which are the mean functional image, the whole-brain structural image, the study-specific template, and the MNI template and using these skull-stripped images as inputs can rectify the problem, given that the skull-stripping was done properly without any remnants of the skulls or excessively cut tissues.

2.4. Suggestions for reporting registration and normalisation precision for functional LC imaging

Following our outline of an analysis pipeline and a set of quality checks on the precision of spatial transformations for functional and structural LC imaging, we propose the following standards for reporting group-level LC imaging results. Many existing publications already include information on some of the aspects outlined below. However, very few encompass all aspects outlined here, especially when it comes to including information on quality assessments of spatial transformations. As shown in Figure 1, our ability to reliably identify LC activations at the group level is crucially dependent on the precision of the post-hoc spatial transformations of the functional images. Thus, we would like to propose the following information to be included as a reporting standard of LC imaging studies:

(1) To assess precision in spatial transformations of functional LC data: In-plane distances of landmarks drawn on each subject's mean functional images in the MNI space to pre-defined landmarks on the structural MNI or group template image (cf. Figure 3).

(2) To assess precision in spatial transformations of structural LC data: Slice-wise distances between the centre of an LC template mask and the centre of each subject's LC mask in MNI or group template space (cf. Figure 4).

Moreover, given the LCs' position in the brainstem and its small size, the following information on data preprocessing should be given:

(3) The description of the movement correction method should include replicable details and should mention any deviations from default settings or additional correctional techniques performed.

(4) The description of the physiological noise correction method should include replicable details and mention any deviations from default settings or additional correctional approaches taken. If no physiological parameters have been recorded, independent component analysis (ICA) approaches can be used to achieve similar effects (Beckmann & Smith, 2004).

3. Codes

3.1. The spatial transformation pipeline (a shell script)

```
#!/bin/bash

# subject ID
ID=1001

# set up folders and group space images
folder=/mnt/work/temp/ED_coreg/"${ID}"/
MNI=/mnt/work/temp/ED_coreg/mni_icbm152_t1_tal_nlin_asym_09c.nii
template=/mnt/work/temp/ED_coreg/pilot_template.nii.gz

# LC segmentation
LCmask=$(ls -t "${folder}"/data/LCmask_"${ID}".nii.gz)

# ---------- prepare images for trasnformation ---------- #

# bias field correct T1 and EPI
N4BiasFieldCorrection -d 3 -v 1 -r 0 -i "${folder}"data/T1mean.nii -o
"${folder}"data/T1mean_corrected.nii -s 2 -c [200x150x100x50,1e-6] -b 200
N4BiasFieldCorrection -d 3 -v 1 -r 0 -i "${folder}"data/meanEPI.nii -o
"${folder}"data/meanEPI_corrected.nii -s 2 -c [200x150x100x50,1e-6] -b 200

# make an EPI mask (FSL)
/usr/local/fsl/bin/bet "${folder}"data/meanEPI_corrected.nii "${folder}"data/meanEPI_brain -f
0.5 -g 0 -n -m

# resample t1slab to T1 resolution (FreeSurfer)
mri_convert -cs 1 -odt float -rl "${folder}"data/T1mean.nii -rt cubic "${folder}"data/t1slab.nii
"${folder}"data/t1slab_1mm.nii

# ------------------------------------------------------- #
```

```
# ---------- start the transformation ---------- #

# study template -> MNI
antsRegistrationSyN.sh -d 3 -t s -f "${MNI}" -m "${template}" -o
"${folder}"NLreg_template_to_MNI_

# T1 -> study template
antsRegistrationSyN.sh -d 3 -t s -f "${template}" -m "${folder}"data/T1mean_corrected.nii -o
"${folder}"data/NLreg_T1mean_to_template_

# T1 -> EPI
antsRegistrationSyN.sh -d 3 -t r -m "${folder}"data/T1mean_corrected.nii -f
"${folder}"data/meanEPI_corrected.nii -x "${folder}"data/meanEPI_brain_mask.nii.gz -o
"${folder}"data/coreg_T1mean_to_meanEPI_

# t1slab(resampled) -> T1
antsRegistrationSyN.sh -d 3 -t r -m "${folder}"data/t1slab_1mm.nii -f
"${folder}"data/T1mean_corrected.nii -o "${folder}"data/coreg_t1slab_to_T1mean_

# T1 -> MNI
antsApplyTransforms -d 3 -v 0 -n BSpline[4] -t
"${folder}"NLreg_template_to_MNI_1Warp.nii.gz -t
"${folder}"NLreg_template_to_MNI_0GenericAffine.mat -t
"${folder}"data/NLreg_T1mean_to_template_1Warp.nii.gz -t
"${folder}"data/NLreg_T1mean_to_template_0GenericAffine.mat -i
"${folder}"data/T1mean_corrected.nii -r "${MNI}" -o
"${folder}"data/NLreg_T1mean_to_MNI.nii

# EPI -> MNI
antsApplyTransforms -d 3 -v 0 -n BSpline[4] -t
"${folder}"NLreg_template_to_MNI_1Warp.nii.gz -t
"${folder}"NLreg_template_to_MNI_0GenericAffine.mat -t
"${folder}"data/NLreg_T1mean_to_template_1Warp.nii.gz -t
"${folder}"data/NLreg_T1mean_to_template_0GenericAffine.mat -t
["${folder}"data/coreg_T1mean_to_meanEPI_0GenericAffine.mat, 1] -i
"${folder}"data/meanEPI_corrected.nii -r "${MNI}" -o
"${folder}"data/NLreg_meanEPI_to_MNI.nii
```

```
# LC segmentation -> MNI
antsApplyTransforms -d 3 -v 0 -n NearestNeighbor -t
"${folder}"NLreg_template_to_MNI_1Warp.nii.gz -t
"${folder}"/NLreg_template_to_MNI_0GenericAffine.mat -t
"${folder}"/data/NLreg_T1mean_to_template_1Warp.nii.gz -t
"${folder}"/data/NLreg_T1mean_to_template_0GenericAffine.mat -t
"${folder}"/data/coreg_t1slab_to_T1mean_0GenericAffine.mat -i "${LCmask}" -r "${MNI}" -o
"${folder}"/data/NLreg_LCmask_to_MNI.nii

# 1st-level stats images -> MNI : ! linear interpolation !
for I in {01..19}
do
        antsApplyTransforms -d 3 -v 0 -n Linear -t
"${folder}"NLreg_template_to_MNI_1Warp.nii.gz -t
"${folder}"NLreg_template_to_MNI_0GenericAffine.mat -t
"${folder}"data/NLreg_T1mean_to_template_1Warp.nii.gz -t
"${folder}"data/NLreg_T1mean_to_template_0GenericAffine.mat -t
["${folder}"data/coreg_T1mean_to_meanEPI_0GenericAffine.mat, 1] -i
"${folder}"data/con_00${I}.nii -r "${MNI}" -o "${folder}"data/con_00${I}_mni.nii
done

# ------------------------------------------------------- #
```

3.2. Generate heatmap and binary images of the aggregated landmarks and transformed LC segmentations in the MNI space

```
%% Check the EPI spatial transformation with landmarks on the transformed
EPI

clear;clc

% preparation

% SET PATHS
path_root = '/path/to/your/landmark/images/';
path_save = '/path/to/the/folder/where/the/heatmap/will/be/saved/';
cd(path_save)

% load IDs
load('/path/to/your/IDs.mat')

% binary or heatmap?
prompt = {'Which image will you generate?: binary or heatmap'};
dlgtitle = 'Input';
definput = {'heatmap'};
dims = [1 35];
answer=inputdlg(prompt,dlgtitle,dims,definput);
if strcmp(answer{1,1},'heatmap')
binarise=0;
elseif strcmp(answer{1,1},'binary')
binarise=1;
end

%% make

ccx = 0;ccy = 0;ccz = 0; % reset the counter
MNI =
spm_read_vols(spm_vol(['/Users/alex/Dropbox/Masks/MNI/mni_icbm152_t1_tal_nl
in_asym_09c.nii'])); % change here accordingly to your file name
averageplot = zeros(size(MNI)); % pre-assign the dummy variable for
plotting the masks. must be the same size with the group/MNI image

for id = 1:length(ids)

    averplotdummy = zeros(size(MNI)); % an empty matrix that landmarks will
be drawn

    name = ids{id};
    data = spm_read_vols(spm_vol([path_root name
'/data/EPILandmark_mni.nii'])); % change here accordingly to your file name
    [x y z] = ndgrid(1:size(data,1), 1:size(data,2), 1:size(data,3));

    % get x/y/z level of landmarks
    ycoordz = y(find(data(:) ~= 0)); uycoordz = unique(ycoordz);
    ylevel=uycoordz;

    xcoordz = x(find(data(:) ~= 0)); uxcoordz = unique(xcoordz);
    xlevel=uxcoordz;

    zcoordz = z(find(data(:) ~= 0)); uzcoordz = unique(zcoordz);
    zlevel=uzcoordz;
```

```
    ccz = ccz+1;
    for zc=1:length(zlevel)
        dumz = squeeze(data(:,:,zlevel(zc)));
        averplotdummy(:,:,zlevel(zc)) = dumz; % average points and squeeze
them in
    end

    ccy = ccy+1;
    for yc=1:length(ylevel)
        dumy = squeeze(data(:,:,ylevel(yc)));
        averplotdummy(:,:,ylevel(yc)) = dumy; % average points and squeeze
them in
    end

    ccx = ccx+1;
    for xc=1:length(xlevel)
        dumx = squeeze(data(:,:,xlevel(xc)));
        averplotdummy(:,:,xlevel(xc)) = dumx; % average points and squeeze
them in
    end

    averageplot = averageplot + averplotdummy; % record the points
    clear dumx dumy dumz

end


%% generate the overlay image in 3D space

if binarise==0
averplot_nifti = averageplot./length(ids);
elseif binarise==1
averplot_nifti = averageplot;
averplot_nifti(find(averplot_nifti(:)>0)) = 1;
end
hdr = spm_vol([path_root ids{1} '/data/EPILandmark_mni.nii']); % pick just
any header from a file
hdr.fname = [path_save 'heatmap_landmarks.nii'];
hdr.dim = size(averplot_nifti);
hdr = rmfield(hdr,'pinfo');
hdr.nii = spm_write_vol(hdr,averplot_nifti);

%% Check the structural transformation with the transformed LC
segmentations

% preparation

% SET PATHS
path_root = '/path/to/your/landmark/images/';
path_save = '/path/to/the/folder/where/the/heatmap/will/be/saved/';
cd(path_save)

% load IDs
load('/path/to/your/IDs.mat')

% binary or heatmap?
prompt = {'Which image will you generate?: binary or heatmap'};
dlgtitle = 'Input';
definput = {'heatmap'};
dims = [1 35];
```

```
answer=inputdlg(prompt,dlgtitle,dims,definput);
if strcmp(answer{1,1},'heatmap')
binarise=0;
elseif strcmp(answer{1,1},'binary')
binarise=1;
end

%% make

MNI =
spm_read_vols(spm_vol(['/Users/alex/Dropbox/Masks/MNI/mni_icbm152_t1_tal_nl
in_asym_09c.nii'])); % change here accordingly to your file name
averageplot = zeros(size(MNI)); % pre-assign the dummy variable for
plotting the masks. must be the same size with the group/MNI image

for id = 1:length(ids)

    averplotdummy = zeros(size(MNI));
    name = ids{id};
    data = spm_read_vols(spm_vol([path_root name '_conjMask_NN.nii'])); %
change here accordingly to your file name
    [x y z] = ndgrid(1:size(data,1), 1:size(data,2), 1:size(data,3));

    % get x/y/z level of landmarks
    ycoordz = y(find(data(:) ~= 0)); uycoordz = unique(ycoordz);
    ylevel=uycoordz;

    xcoordz = x(find(data(:) ~= 0)); uxcoordz = unique(xcoordz);
    xlevel=uxcoordz;

    zcoordz = z(find(data(:) ~= 0)); uzcoordz = unique(zcoordz);
    zlevel=uzcoordz;

    for zc=1:length(zlevel)
        dumz = squeeze(data(:,:,zlevel(zc)));
        averplotdummy(:,:,zlevel(zc)) = dumz; % average points and squeeze
them in
    end

    for yc=1:length(ylevel)
        dumy = squeeze(data(:,:,ylevel(yc)));
        averplotdummy(:,:,ylevel(yc)) = dumy; % average points and squeeze
them in
    end

    for xc=1:length(xlevel)
        dumx = squeeze(data(:,:,xlevel(xc)));
        averplotdummy(:,:,xlevel(xc)) = dumx; % average points and squeeze
them in
    end

    averplotdummy(find(averplotdummy(:)>0)) = 1;

    averageplot = averageplot + averplotdummy; % record the points

    clear dumx dumy dumz

end

%% generate the overlay image in 3D space
```

```
if binarise==0
averplot_nifti = averageplot./length(ids);
elseif binarise==1
averplot_nifti = averageplot;
averplot_nifti(find(averplot_nifti(:)>0)) = 1;
end
hdr = spm_vol([path_root ids{1} '_conjMask_NN.nii']); % pick just any
header from a file
hdr.fname = [path_save 'heatmap_segmentations.nii'];
hdr.dim = size(averplot_nifti);
hdr = rmfield(hdr,'pinfo');
hdr.nii = spm_write_vol(hdr,averplot_nifti);
```

3.3. Generate video of transformed mean functional or structural images (a MATLAB script)

```
%%  Visual check for transformed images
%   make a movie of transformed EPIs/T1s to see whether there's any obvious
%   misalignment

%% preparation

clear;clc

% set paths
path_root = '/path/to/your/transformed/images/'; % where are the files?
path_save = '/path/where/the/video/will/be/saved/'; % where is the video
going to be saved?

% set variables
ids = [];

input_prompt = {'duration per frame in seconds (default=0)'; 'name of the
file'; 'view(sagittal, coronal, axial)'; 'which slice should the video
capture?'};
defaults     = {'0','null','sagittal','96'};
input_answer = inputdlg(input_prompt, 'specify the properties of the video',
1, defaults);

vidframerate = str2num(input_answer{1,1});
vidformat    = '.avi';
vidname      = [input_answer{2,1} vidformat];
vidview      = input_answer{3,1};
slicenum     = str2num(input_answer{4,1});

cd(path_save)
v            = VideoWriter(vidname);
if vidframerate ~= 0
    v.FrameRate  = 1/vidframerate;
elseif vidframerate == 0
    v.FrameRate  = 30; % it's super fast
end

%% start recording
```

```
open(v) % open the file

cc = 0; figure % open a sketchbook
for i = 1:length(ids)
        name = num2str(ids(i)); disp(num2str(ids(i)))
        data        =        spm_read_vols(spm_vol([path_root        name
'/data/transformed_image.nii']));

        % position the slice - dimensions are: (x=sagittal, y=coronal,
z=axial)
        if strcmpi(vidview,'sagittal')
            dum = squeeze(data(slicenum,:,:)); % sagittal
        elseif strcmpi(vidview,'coronal')
            dum = squeeze(data(:,slicenum,:)); % coronal view
        elseif strcmpi(vidview,'axial')
            dum = squeeze(data(:,:,slicenum)); % axial
        else
            error('check your view input')
        end

        dum = rot90(dum);% t1WB on template
%     dum = zscore(dum);

        imagesc(dum);
        title(num2str(ids(i)))
        cc = cc+1;
        % Store the frame
        M(cc)=getframe(gcf); % leaving gcf out crops the frame in the movie.

end

writeVideo(v,M) % export the video
close(v)
close all;
```

3.4. Calculate the in-plane distance of single-subject landmarks (a MATLAB script)

3.4.1. When the landmarks were evaluated with size-3 spherical segmentation pen

```
%% Calculate the in-plane distances


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% use this script when using size-3-spherical seg pen instead of 1-voxel
pen
%                              ^^^^^^^^^^^^^^^^
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc;clear;close all

% who rated the images?
raterLabel=inputdlg('Who evaluated the landmarks? Type initials without
spaces (e.g. AY)');

% set env
path_transformed =
'/path/to/your/transformed/mean/functional/MRI/landmarks/';
load('/load/list/of/IDs/in/cell/structure/ID.mat')

% load individual landmark images
transformed_landmarks_single=[];
for subj=1:length(IDs)
    transformed_landmarks_single{subj,1}=spm_read_vols(spm_vol([
path_transformed IDs{subj} '_landmark.nii']));
end; clear subj

% load the landmark image drawn on MNI
MNIlandmark = spm_read_vols(spm_vol(['/path/to/your/MNI-
landmarks/MNI_landmarks_for_EPI.nii'])); % change here accordingly to your
file name
MNIlandmark(find(MNIlandmark<1))=NaN;

% write down the coordinates of MNI landmarks
[x_mni,y_mni,z_mni] = ind2sub(size(MNIlandmark),find(~isnan(MNIlandmark)));
tmp = [x_mni,y_mni,z_mni];
[~,idx1] = sort(tmp(:,3)); tmp_sorted = tmp(idx1,:);

% sort the coordinates by slices
slices = unique(tmp(:,3));
slices = slices(end:-1:1);

% the first slice, with three landmarks: top, NucRubs
clear idx1 positionSlice positionLR_sorted cluster1 cluster2 cluster3
TopSliceClusters ordersLR

positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(1)),:); % find the
points in the first slice
[~,idx1] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx1,:);
TopSliceClusters = kmeans(positionLR_sorted(:,1),3);
```

```
ordersLR=unique(TopSliceClusters,'stable');

% ruber right
cluster1 = positionLR_sorted(TopSliceClusters==ordersLR(3),:);
MNI.NucRuber_right = [median(cluster1(:,1)) median(cluster1(:,2))
slices(1)]; %ceil( median(cluster3) );

% top curve
cluster2 = positionLR_sorted(TopSliceClusters==ordersLR(2),:);
MNI.TopBrainstem = [median(cluster2(:,1)) median(cluster2(:,2)) slices(1)];
%ceil( median(cluster3) );

% ruber left
cluster3 = positionLR_sorted(TopSliceClusters==ordersLR(1),:);
MNI.NucRuber_left = [median(cluster3(:,1)) median(cluster3(:,2))
slices(1)]; %ceil( median(cluster3) );

% the second slice, with four landmarks: brainstem outline and LC

clear idx1 positionSlice positionLR_sorted cluster1 cluster2 cluster3
MiddleSliceClusters ordersLR

positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(2)),:);
[~,idx1] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx1,:);
MiddleSliceClusters = kmeans(positionLR_sorted(:,1),4);
ordersLR=unique(MiddleSliceClusters,'stable'); % identify cluster indice

% brainstem outline right
cluster1 = positionLR_sorted(MiddleSliceClusters==ordersLR(4),:);
MNI.OutlineBrainstem_right = [median(cluster1(:,1)) median(cluster1(:,2))
slices(2)]; %ceil( median(cluster1) );

% LC right
cluster2 = positionLR_sorted(MiddleSliceClusters==ordersLR(3),:);
MNI.LC_right = [median(cluster2(:,1)) median(cluster2(:,2)) slices(2)];

% LC left
cluster3 = positionLR_sorted(MiddleSliceClusters==ordersLR(2),:);
MNI.LC_left = [median(cluster3(:,1)) median(cluster3(:,2)) slices(2)];

% brainstem outline left
cluster4 = positionLR_sorted(MiddleSliceClusters==ordersLR(1),:);
MNI.OutlineBrainstem_left = [median(cluster4(:,1)) median(cluster4(:,2))
slices(2)];


% the third slice, with one landmark: bottom window brainstem

clear idx1 positionSlice positionLR_sorted cluster1 cluster2 cluster3
BottomSliceClusters

positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(3)),:);
[~,idx1] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx1,:);

% brainstem bottom
MNI.BottomBrainstem = [median(positionLR_sorted(:,1))
median(positionLR_sorted(:,2)) slices(3)]; %ceil( median(positionLR_sorted)
);
```

```
disp('prep done')

%% identify median coords in the single landmarks

transformed_landmark_coords=[];

for subj=1:length(IDs)

    clear xs ys zs tmp idx2

    [xs,ys,zs] =
ind2sub(size(transformed_landmarks_single{subj}),find(transformed_landmarks
_single{subj}~=0));
    tmp = [xs ys zs];
    [~,idx2] = sort(tmp(:,3)); tmp_sorted = tmp(idx2,:);

    slices = unique(tmp(:,3));
    slices = slices(end:-1:1);

    %% the first slice, with three landmarks: top, NucRubs

    clear idx3 positionSlice positionLR_sorted cluster1 cluster2 cluster3
TopSliceClusters ordersLR

    transformed_landmark_coords{subj,1}.TopSlice = [];

    positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(1)),:); % find
the points in the first slice
    [~,idx3] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx3,:);
    TopSliceClusters = kmeans(positionLR_sorted(:,1),3);
    ordersLR=unique(TopSliceClusters,'stable');

    % ruber right
    cluster1 = positionLR_sorted(TopSliceClusters==ordersLR(3),:);
    transformed_landmark_coords{subj,1}.TopSlice.NucRuber_right =
[median(cluster1(:,1)) median(cluster1(:,2)) slices(1)]; %ceil(
median(cluster3) );

    % top curve
    cluster2 = positionLR_sorted(TopSliceClusters==ordersLR(2),:);
    transformed_landmark_coords{subj,1}.TopSlice.TopBrainstem =
[median(cluster2(:,1)) median(cluster2(:,2)) slices(1)]; %ceil(
median(cluster3) );

    % ruber left
    cluster3 = positionLR_sorted(TopSliceClusters==ordersLR(1),:);
    transformed_landmark_coords{subj,1}.TopSlice.NucRuber_left =
[median(cluster3(:,1)) median(cluster3(:,2)) slices(1)]; %ceil(
median(cluster3) );

    %% the second slice, with four landmarks: brainstem outline and LC

    clear idx3 positionSlice positionLR_sorted cluster1 cluster2 cluster3
MiddleSliceClusters ordersLR

    transformed_landmark_coords{subj,1}.MidSlice = [];
```

```
    positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(2)),:);
    [~,idx3] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx3,:);
    MiddleSliceClusters = kmeans(positionLR_sorted(:,1),4);
    ordersLR=unique(MiddleSliceClusters,'stable'); % identify cluster
indice

    % brainstem outline right
    cluster1 = positionLR_sorted(MiddleSliceClusters==ordersLR(4),:);
    transformed_landmark_coords{subj,1}.MidSlice.OutlineBrainstem_right =
[median(cluster1(:,1)) median(cluster1(:,2)) slices(2)]; %ceil(
median(cluster1) );

    % LC right
    cluster2 = positionLR_sorted(MiddleSliceClusters==ordersLR(3),:);
    transformed_landmark_coords{subj,1}.MidSlice.LC_right =
[median(cluster2(:,1)) median(cluster2(:,2)) slices(2)];

    % LC left
    cluster3 = positionLR_sorted(MiddleSliceClusters==ordersLR(2),:);
    transformed_landmark_coords{subj,1}.MidSlice.LC_left =
[median(cluster3(:,1)) median(cluster3(:,2)) slices(2)];

    % brainstem outline left
    cluster4 = positionLR_sorted(MiddleSliceClusters==ordersLR(1),:);
    transformed_landmark_coords{subj,1}.MidSlice.OutlineBrainstem_left =
[median(cluster4(:,1)) median(cluster4(:,2)) slices(2)];


    %% the third slice, with one landmark: bottom window brainstem

    clear idx3 positionSlice positionLR_sorted cluster1 cluster2 cluster3
BottomSliceClusters

    transformed_landmark_coords{subj,1}.BottomSlice = [];

    positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(3)),:);
    [~,idx3] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx3,:);

    % brainstem bottom
    transformed_landmark_coords{subj,1}.BottomSlice.BottomBrainstem =
[median(positionLR_sorted(:,1)) median(positionLR_sorted(:,2)) slices(3)];
%ceil( median(positionLR_sorted) );

    fprintf('\n subject %s done\n',IDs{subj})

end;clear subj

disp('coordinates collected')

%% calculate distances in single subject level

varnames =
{'NucRuber_left';'NucRuber_right';'TopBrainstem';'OutlineBrainstem_left';'O
utlineBrainstem_right';...
    'LC_left';'LC_right';'BottomBrainstem'};
for v1=1:length(varnames)
   eval([varnames{v1} '=[];'])
end
```

```
Distances_indv=[];
for subj=1:length(IDs)

    Distances_indv{subj,1}.NucRuber=[];
    Distances_indv{subj,1}.NucRuber.left =
sum((transformed_landmark_coords{subj,1}.TopSlice.NucRuber_left-
MNI.NucRuber_left).^2).^0.5;
    Distances_indv{subj,1}.NucRuber.right =
sum((transformed_landmark_coords{subj,1}.TopSlice.NucRuber_right-
MNI.NucRuber_right).^2).^0.5;

    Distances_indv{subj,1}.TopBrainstem =
sum((transformed_landmark_coords{subj,1}.TopSlice.TopBrainstem-
MNI.TopBrainstem).^2).^0.5;

    Distances_indv{subj,1}.OutlineBrainstem=[];

Distances_indv{subj,1}.OutlineBrainstem.left=sum((transformed_landmark_coor
ds{subj,1}.MidSlice.OutlineBrainstem_left-
MNI.OutlineBrainstem_left).^2).^0.5;

Distances_indv{subj,1}.OutlineBrainstem.right=sum((transformed_landmark_coo
rds{subj,1}.MidSlice.OutlineBrainstem_right-
MNI.OutlineBrainstem_right).^2).^0.5;

    Distances_indv{subj,1}.LC=[];

Distances_indv{subj,1}.LC.left=sum((transformed_landmark_coords{subj,1}.Mid
Slice.LC_left-MNI.LC_left).^2).^0.5;

Distances_indv{subj,1}.LC.right=sum((transformed_landmark_coords{subj,1}.Mi
dSlice.LC_right-MNI.LC_right).^2).^0.5;

    Distances_indv{subj,1}.BottomBrainstem =
sum((transformed_landmark_coords{subj,1}.BottomSlice.BottomBrainstem-
MNI.BottomBrainstem).^2).^0.5;

    % write as a table
    NucRuber_left(subj,1)=Distances_indv{subj,1}.NucRuber.left;
    NucRuber_right(subj,1)=Distances_indv{subj,1}.NucRuber.right;
    TopBrainstem(subj,1)=Distances_indv{subj,1}.TopBrainstem;

OutlineBrainstem_left(subj,1)=Distances_indv{subj,1}.OutlineBrainstem.left;

OutlineBrainstem_right(subj,1)=Distances_indv{subj,1}.OutlineBrainstem.righ
t;
    LC_left(subj,1)=Distances_indv{subj,1}.LC.left;
    LC_right(subj,1)=Distances_indv{subj,1}.LC.right;
    BottomBrainstem(subj,1)=Distances_indv{subj,1}.BottomBrainstem;

end

IDcolumn = cell2mat(cellfun(@str2num, IDs,'UniformOutput',false));

T=table(NucRuber_left,NucRuber_right,TopBrainstem,OutlineBrainstem_left,Out
lineBrainstem_right,...
    LC_left,LC_right,BottomBrainstem);
Distance_export
=[IDcolumn',NucRuber_left,NucRuber_right,TopBrainstem,OutlineBrainstem_left
,OutlineBrainstem_right,...
```

```
    LC_left,LC_right,BottomBrainstem];

save([path_transformed 'Distance_EPILandmarks_' raterLabel
'.mat'],'Distance_export')
save([path_transformed 'LandmarkCoordinates_' raterLabel
'.mat'],'transformed_landmark_coords')

disp('distance calc done')

%% calculate interrator agreement

load('/load/list/of/IDs/in/cell/structure/ID.mat') % load IDs of rated
images

clear transformed_landmark_coords
rater1=load('/load/the/coordinates/of/landmarks/LandmarkCoordinates_rater1.
mat');
rater2=load('/load/the/coordinates/of/landmarks/LandmarkCoordinates_rater2.
mat');

% calculate agreements per subject per landmark
IR=[];
for id=1:length(IDs)


    % top slice
    clear rater1dat rater2dat
    rater1dat=rater1.transformed_landmark_coords{id,1}.TopSlice;
    rater2dat=rater2.transformed_landmark_coords{id,1}.TopSlice;

    IR{id,1}.NucRuber_left
=sum(rater1dat.NucRuber_left(1:2)==rater2dat.NucRuber_left(1:2));
    IR{id,1}.NucRuber_right
=sum(rater1dat.NucRuber_right(1:2)==rater2dat.NucRuber_right(1:2));
    IR{id,1}.TopBrainstem
=sum(rater1dat.TopBrainstem(1:2)==rater2dat.TopBrainstem(1:2));

    NucRuber_left(id,1)
=sum(rater1dat.NucRuber_left(1:2)==rater2dat.NucRuber_left(1:2));
    NucRuber_right(id,1)
=sum(rater1dat.NucRuber_right(1:2)==rater2dat.NucRuber_right(1:2));
    TopBrainstem(id,1)
=sum(rater1dat.TopBrainstem(1:2)==rater2dat.TopBrainstem(1:2));


    cRater1.NucRuber_left(id,:)         = rater1dat.NucRuber_left;
    cRater1.NucRuber_right(id,:)        = rater1dat.NucRuber_right;
    cRater1.TopBrainstem(id,:)          = rater1dat.TopBrainstem;

    cRater2.NucRuber_left(id,:)         = rater2dat.NucRuber_left;
    cRater2.NucRuber_right(id,:)        = rater2dat.NucRuber_right;
    cRater2.TopBrainstem(id,:)          = rater2dat.TopBrainstem;


    % middle slice
    clear ay ml
    rater1dat=rater1.transformed_landmark_coords{id,1}.MidSlice;
    rater2dat=rater2.transformed_landmark_coords{id,1}.MidSlice;
```

```
    IR{id,1}.OutlineBrainstem_left
=sum(rater1dat.OutlineBrainstem_left(1:2)==rater2dat.OutlineBrainstem_left(
1:2));
    IR{id,1}.OutlineBrainstem_right
=sum(rater1dat.OutlineBrainstem_right(1:2)==rater2dat.OutlineBrainstem_righ
t(1:2));
    IR{id,1}.LC_left
=sum(rater1dat.LC_left(1:2)==rater2dat.LC_left(1:2));
    IR{id,1}.LC_right
=sum(rater1dat.LC_right(1:2)==rater2dat.LC_right(1:2));

    OutlineBrainstem_left(id,1)
=sum(rater1dat.OutlineBrainstem_left(1:2)==rater2dat.OutlineBrainstem_left(
1:2));
    OutlineBrainstem_right(id,1)
=sum(rater1dat.OutlineBrainstem_right(1:2)==rater2dat.OutlineBrainstem_righ
t(1:2));
    LC_left(id,1)
=sum(rater1dat.LC_left(1:2)==rater2dat.LC_left(1:2));
    LC_right(id,1)
=sum(rater1dat.LC_right(1:2)==rater2dat.LC_right(1:2));


    cRater1.OutlineBrainstem_left(id,:) = rater1dat.OutlineBrainstem_left;
    cRater1.OutlineBrainstem_right(id,:)= rater1dat.OutlineBrainstem_right;
    cRater1.LC_left(id,:)               = rater1dat.LC_left;
    cRater1.LC_right(id,:)              = rater1dat.LC_right;

    cRater2.OutlineBrainstem_left(id,:) = rater2dat.OutlineBrainstem_left;
    cRater2.OutlineBrainstem_right(id,:)= rater2dat.OutlineBrainstem_right;
    cRater2.LC_left(id,:)               = rater2dat.LC_left;
    cRater2.LC_right(id,:)              = rater2dat.LC_right;


    % bottom slice
    clear ay ml
    rater1dat=rater1.transformed_landmark_coords{id,1}.BottomSlice;
    rater2dat=rater2.transformed_landmark_coords{id,1}.BottomSlice;
    IR{id,1}.BottomBrainstem
=sum(rater1dat.BottomBrainstem(1:2)==rater2dat.BottomBrainstem(1:2));
    BottomBrainstem(id,1)
=sum(rater1dat.BottomBrainstem(1:2)==rater2dat.BottomBrainstem(1:2));


    cRater1.BottomBrainstem(id,:)       = rater1dat.BottomBrainstem;
    cRater2.BottomBrainstem(id,:)       = rater2dat.BottomBrainstem;

end

% calculate DICE score

DICE=[];

DICE.NucRuber_left=(sum(NucRuber_left)*2)/(2*2*numel(IDs));
DICE.NucRuber_right=(sum(NucRuber_right)*2)/(2*2*numel(IDs));
DICE.TopBrainstem=(sum(TopBrainstem)*2)/(2*2*numel(IDs));

DICE.OutlineBrainstem_left=(sum(OutlineBrainstem_left)*2)/(2*2*numel(IDs));
DICE.OutlineBrainstem_right=(sum(OutlineBrainstem_right)*2)/(2*2*numel(IDs)
);
```

```
DICE.LC_left=(sum(LC_left)*2)/(2*2*numel(IDs));
DICE.LC_right=(sum(LC_right)*2)/(2*2*numel(IDs));

DICE.BottomBrainstem=(sum(BottomBrainstem)*2)/(2*2*numel(IDs));
```

3.4.2. When the landmarks were evaluated with size-1 spherical segmentation pen

```
%% Calculate the in-plane distances


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% use this script when using 1-voxel seg pen instead of 3-voxel spherical
pen
%                             ^^^^^^^
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc;clear;close all

% who rated the images?
raterLabel=inputdlg('Who evaluated the landmarks? Type initials without
spaces (e.g. AY)');

% set env
path_transformed =
'/path/to/your/transformed/mean/functional/MRI/landmarks/';
load('/load/list/of/IDs/in/cell/structure/ID.mat')

% load individual landmark images
transformed_landmarks_single=[];
for subj=1:length(IDs)
    transformed_landmarks_single{subj,1}=spm_read_vols(spm_vol([
path_transformed IDs{subj} '_landmark.nii']));
end; clear subj

% load the landmark image drawn on MNI
MNIlandmark = spm_read_vols(spm_vol(['/path/to/your/MNI-
landmarks/MNI_landmarks_for_EPI.nii'])); % change here accordingly to your
file name
MNIlandmark(find(MNIlandmark<1))=NaN;

% write down the coordinates of MNI landmarks
[x_mni,y_mni,z_mni] = ind2sub(size(MNIlandmark),find(~isnan(MNIlandmark)));
tmp = [x_mni,y_mni,z_mni];
[~,idx1] = sort(tmp(:,3)); tmp_sorted = tmp(idx1,:);

% sort the coordinates by slices
slices = unique(tmp(:,3));
slices = slices(end:-1:1);

% the first slice, with three landmarks: top, NucRubs
clear idx1 positionSlice positionLR_sorted cluster1 cluster2 cluster3
TopSliceClusters ordersLR

positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(1)),:); % find the
points in the first slice
[~,idx1] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx1,:);
TopSliceClusters = kmeans(positionLR_sorted(:,1),3);
ordersLR=unique(TopSliceClusters,'stable');
```

```
% ruber right
cluster1 = positionLR_sorted(TopSliceClusters==ordersLR(3),:);
MNI.NucRuber_right = [median(cluster1(:,1)) median(cluster1(:,2))
slices(1)]; %ceil( median(cluster3) );

% top curve
cluster2 = positionLR_sorted(TopSliceClusters==ordersLR(2),:);
MNI.TopBrainstem = [median(cluster2(:,1)) median(cluster2(:,2)) slices(1)];
%ceil( median(cluster3) );

% ruber left
cluster3 = positionLR_sorted(TopSliceClusters==ordersLR(1),:);
MNI.NucRuber_left = [median(cluster3(:,1)) median(cluster3(:,2))
slices(1)]; %ceil( median(cluster3) );

% the second slice, with four landmarks: brainstem outline and LC

clear idx1 positionSlice positionLR_sorted cluster1 cluster2 cluster3
MiddleSliceClusters ordersLR

positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(2)),:);
[~,idx1] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx1,:);
MiddleSliceClusters = kmeans(positionLR_sorted(:,1),4);
ordersLR=unique(MiddleSliceClusters,'stable'); % identify cluster indice

% brainstem outline right
cluster1 = positionLR_sorted(MiddleSliceClusters==ordersLR(4),:);
MNI.OutlineBrainstem_right = [median(cluster1(:,1)) median(cluster1(:,2))
slices(2)]; %ceil( median(cluster1) );

% LC right
cluster2 = positionLR_sorted(MiddleSliceClusters==ordersLR(3),:);
MNI.LC_right = [median(cluster2(:,1)) median(cluster2(:,2)) slices(2)];

% LC left
cluster3 = positionLR_sorted(MiddleSliceClusters==ordersLR(2),:);
MNI.LC_left = [median(cluster3(:,1)) median(cluster3(:,2)) slices(2)];

% brainstem outline left
cluster4 = positionLR_sorted(MiddleSliceClusters==ordersLR(1),:);
MNI.OutlineBrainstem_left = [median(cluster4(:,1)) median(cluster4(:,2))
slices(2)];


% the third slice, with one landmark: bottom window brainstem

clear idx1 positionSlice positionLR_sorted cluster1 cluster2 cluster3
BottomSliceClusters

positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(3)),:);
[~,idx1] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx1,:);

% brainstem bottom
MNI.BottomBrainstem = [median(positionLR_sorted(:,1))
median(positionLR_sorted(:,2)) slices(3)]; %ceil( median(positionLR_sorted)
);

disp('prep done')
```

```
%% identify median coords in the single landmarks

transformed_landmark_coords=[];

for subj=1:length(IDs)

    clear xs ys zs tmp idx2

    [xs,ys,zs] =
ind2sub(size(transformed_landmarks_single{subj}),find(transformed_landmarks
_single{subj}~=0)); % index the coordinates
    tmp = [xs ys zs]; % 2-dim matrix
    [~,idx2] = sort(tmp(:,3)); tmp_sorted = tmp(idx2,:);

    slices = unique(tmp(:,3)); % sort with slice numbers
    slices = slices(end:-1:1); % order into S -> I direction

    %% the first slice, with three landmarks: top, NucRubs
    clear idx3 positionSlice positionX_sorted

    transformed_landmark_coords{subj,1}.TopSlice = [];

    positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(1)),:); % find
the points in the first slice
    [~,idx3] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx3,:); % sort coordinates in L -> R direction

    % ruber left
    clear r c
    [r,c]=find(positionLR_sorted(:,1)==min(positionLR_sorted(:,1))); % find
the voxel that are the most left
    transformed_landmark_coords{subj,1}.TopSlice.NucRuber_left =
positionLR_sorted(r,:);

    % ruber right
    clear r c
    [r,c]=find(positionLR_sorted(:,1)==max(positionLR_sorted(:,1))); % find
the voxel that are the most right
    transformed_landmark_coords{subj,1}.TopSlice.NucRuber_right =
positionLR_sorted(r,:);

    % top curve
    clear r c

[r,c]=find(positionLR_sorted(:,1)==median(positionLR_sorted(:,1),'all')); %
find the voxel that are in the middle
    transformed_landmark_coords{subj,1}.TopSlice.TopBrainstem =
positionLR_sorted(r,:);


    %% the second slice, with four landmarks: brainstem outline and LC

    clear idx3 idx4 positionY positionX_sorted

    transformed_landmark_coords{subj,1}.MidSlice = [];

    positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(2)),:); % find
the points in the second slice
```

```
    [~,idx3] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx3,:); % sort coordinates in L -> R direction

    % brainstem outline left
    clear r c
    [r,c]=find(positionLR_sorted(:,1)==positionLR_sorted(1,1));
    transformed_landmark_coords{subj,1}.MidSlice.OutlineBrainstem_left =
positionLR_sorted(r,:);

    % LC left
    clear r c
    [r,c]=find(positionLR_sorted(:,1)==positionLR_sorted(2,1));
    transformed_landmark_coords{subj,1}.MidSlice.LC_left =
positionLR_sorted(r,:);

    % LC right
    clear r c
    [r,c]=find(positionLR_sorted(:,1)==positionLR_sorted(3,1));
    transformed_landmark_coords{subj,1}.MidSlice.LC_right =
positionLR_sorted(r,:);

    % brainstem outline right
    clear r c
    [r,c]=find(positionLR_sorted(:,1)==positionLR_sorted(4,1));
    transformed_landmark_coords{subj,1}.MidSlice.OutlineBrainstem_right =
positionLR_sorted(r,:);


    %% the third slice, with one landmark: bottom window brainstem

    clear idx3 positionY positionX_sorted

    transformed_landmark_coords{subj,1}.BottomSlice = [];

    positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(3)),:); % find
the points in the third slice
    [~,idx3] = sort(positionSlice(:,1)); positionLR_sorted =
positionSlice(idx3,:);

    % brainstem bottom
    transformed_landmark_coords{subj,1}.BottomSlice.BottomBrainstem =
positionLR_sorted;

    fprintf('\n subject %s done\n',IDs{subj})

end;clear subj

disp('coordinates collected')

%% calculate distances in single subject level

% flexible listing
varnames =
{'NucRuber_left';'NucRuber_right';'TopBrainstem';'OutlineBrainstem_left';'O
utlineBrainstem_right';...
    'LC_left';'LC_right';'BottomBrainstem'};
for v1=1:length(varnames)
   eval([varnames{v1} '=[];'])
end
```

```
% calculate distances
Distances_indv=[];
for subj=1:length(IDs)

    Distances_indv{subj,1}.NucRuber=[];
    Distances_indv{subj,1}.NucRuber.left =
sum((transformed_landmark_coords{subj,1}.TopSlice.NucRuber_left-
MNI.NucRuber_left).^2).^0.5;
    Distances_indv{subj,1}.NucRuber.right =
sum((transformed_landmark_coords{subj,1}.TopSlice.NucRuber_right-
MNI.NucRuber_right).^2).^0.5;

    Distances_indv{subj,1}.TopBrainstem =
sum((transformed_landmark_coords{subj,1}.TopSlice.TopBrainstem-
MNI.TopBrainstem).^2).^0.5;

    Distances_indv{subj,1}.OutlineBrainstem=[];

Distances_indv{subj,1}.OutlineBrainstem.left=sum((transformed_landmark_coor
ds{subj,1}.MidSlice.OutlineBrainstem_left-
MNI.OutlineBrainstem_left).^2).^0.5;

Distances_indv{subj,1}.OutlineBrainstem.right=sum((transformed_landmark_coo
rds{subj,1}.MidSlice.OutlineBrainstem_right-
MNI.OutlineBrainstem_right).^2).^0.5;

    Distances_indv{subj,1}.LC=[];

Distances_indv{subj,1}.LC.left=sum((transformed_landmark_coords{subj,1}.Mid
Slice.LC_left-MNI.LC_left).^2).^0.5;

Distances_indv{subj,1}.LC.right=sum((transformed_landmark_coords{subj,1}.Mi
dSlice.LC_right-MNI.LC_right).^2).^0.5;

    Distances_indv{subj,1}.BottomBrainstem =
sum((transformed_landmark_coords{subj,1}.BottomSlice.BottomBrainstem-
MNI.BottomBrainstem).^2).^0.5;

    % prepare the variables to write as a table
    NucRuber_left(subj,1)=Distances_indv{subj,1}.NucRuber.left;
    NucRuber_right(subj,1)=Distances_indv{subj,1}.NucRuber.right;
    TopBrainstem(subj,1)=Distances_indv{subj,1}.TopBrainstem;

OutlineBrainstem_left(subj,1)=Distances_indv{subj,1}.OutlineBrainstem.left;

OutlineBrainstem_right(subj,1)=Distances_indv{subj,1}.OutlineBrainstem.righ
t;
    LC_left(subj,1)=Distances_indv{subj,1}.LC.left;
    LC_right(subj,1)=Distances_indv{subj,1}.LC.right;
    BottomBrainstem(subj,1)=Distances_indv{subj,1}.BottomBrainstem;

end


IDcolumn = cell2mat(cellfun(@str2num, IDs,'UniformOutput',false));

T=table(NucRuber_left,NucRuber_right,TopBrainstem,OutlineBrainstem_left,Out
lineBrainstem_right,...
    LC_left,LC_right,BottomBrainstem);
```

```
Distance_export
=[IDcolumn',NucRuber_left,NucRuber_right,TopBrainstem,OutlineBrainstem_left
,OutlineBrainstem_right,...
    LC_left,LC_right,BottomBrainstem];

save([path_transformed 'Distance_EPILandmarks_' raterLabel
'.mat'],'Distance_export')
save([path_transformed 'LandmarkCoordinates_' raterLabel
'.mat'],'transformed_landmark_coords')

disp('distance calc done')

%% calculate interrator agreement

load('/load/list/of/IDs/in/cell/structure/ID.mat') % load IDs of rated
images

clear transformed_landmark_coords
rater1=load('/load/the/coordinates/of/landmarks/LandmarkCoordinates_rater1.
mat');
rater2=load('/load/the/coordinates/of/landmarks/LandmarkCoordinates_rater2.
mat');

% calculate agreements per subject per landmark
IR=[];
for id=1:length(IDs)


    % top slice
    clear rater1dat rater2dat
    rater1dat=rater1.transformed_landmark_coords{id,1}.TopSlice;
    rater2dat=rater2.transformed_landmark_coords{id,1}.TopSlice;

    IR{id,1}.NucRuber_left
=sum(rater1dat.NucRuber_left(1:2)==rater2dat.NucRuber_left(1:2));
    IR{id,1}.NucRuber_right
=sum(rater1dat.NucRuber_right(1:2)==rater2dat.NucRuber_right(1:2));
    IR{id,1}.TopBrainstem
=sum(rater1dat.TopBrainstem(1:2)==rater2dat.TopBrainstem(1:2));

    NucRuber_left(id,1)
=sum(rater1dat.NucRuber_left(1:2)==rater2dat.NucRuber_left(1:2));
    NucRuber_right(id,1)
=sum(rater1dat.NucRuber_right(1:2)==rater2dat.NucRuber_right(1:2));
    TopBrainstem(id,1)
=sum(rater1dat.TopBrainstem(1:2)==rater2dat.TopBrainstem(1:2));


    cRater1.NucRuber_left(id,:)         = rater1dat.NucRuber_left;
    cRater1.NucRuber_right(id,:)        = rater1dat.NucRuber_right;
    cRater1.TopBrainstem(id,:)          = rater1dat.TopBrainstem;

    cRater2.NucRuber_left(id,:)         = rater2dat.NucRuber_left;
    cRater2.NucRuber_right(id,:)        = rater2dat.NucRuber_right;
    cRater2.TopBrainstem(id,:)          = rater2dat.TopBrainstem;


    % middle slice
    clear ay ml
    rater1dat=rater1.transformed_landmark_coords{id,1}.MidSlice;
```

```
    rater2dat=rater2.transformed_landmark_coords{id,1}.MidSlice;
    IR{id,1}.OutlineBrainstem_left
=sum(rater1dat.OutlineBrainstem_left(1:2)==rater2dat.OutlineBrainstem_left(
1:2));
    IR{id,1}.OutlineBrainstem_right
=sum(rater1dat.OutlineBrainstem_right(1:2)==rater2dat.OutlineBrainstem_righ
t(1:2));
    IR{id,1}.LC_left
=sum(rater1dat.LC_left(1:2)==rater2dat.LC_left(1:2));
    IR{id,1}.LC_right
=sum(rater1dat.LC_right(1:2)==rater2dat.LC_right(1:2));

    OutlineBrainstem_left(id,1)
=sum(rater1dat.OutlineBrainstem_left(1:2)==rater2dat.OutlineBrainstem_left(
1:2));
    OutlineBrainstem_right(id,1)
=sum(rater1dat.OutlineBrainstem_right(1:2)==rater2dat.OutlineBrainstem_righ
t(1:2));
    LC_left(id,1)
=sum(rater1dat.LC_left(1:2)==rater2dat.LC_left(1:2));
    LC_right(id,1)
=sum(rater1dat.LC_right(1:2)==rater2dat.LC_right(1:2));


    cRater1.OutlineBrainstem_left(id,:) = rater1dat.OutlineBrainstem_left;
    cRater1.OutlineBrainstem_right(id,:)= rater1dat.OutlineBrainstem_right;
    cRater1.LC_left(id,:)               = rater1dat.LC_left;
    cRater1.LC_right(id,:)              = rater1dat.LC_right;

    cRater2.OutlineBrainstem_left(id,:) = rater2dat.OutlineBrainstem_left;
    cRater2.OutlineBrainstem_right(id,:)= rater2dat.OutlineBrainstem_right;
    cRater2.LC_left(id,:)               = rater2dat.LC_left;
    cRater2.LC_right(id,:)              = rater2dat.LC_right;


    % bottom slice
    clear ay ml
    rater1dat=rater1.transformed_landmark_coords{id,1}.BottomSlice;
    rater2dat=rater2.transformed_landmark_coords{id,1}.BottomSlice;
    IR{id,1}.BottomBrainstem
=sum(rater1dat.BottomBrainstem(1:2)==rater2dat.BottomBrainstem(1:2));
    BottomBrainstem(id,1)
=sum(rater1dat.BottomBrainstem(1:2)==rater2dat.BottomBrainstem(1:2));


    cRater1.BottomBrainstem(id,:)       = rater1dat.BottomBrainstem;
    cRater2.BottomBrainstem(id,:)       = rater2dat.BottomBrainstem;

end

% calculate DICE score

DICE=[];

DICE.NucRuber_left=(sum(NucRuber_left)*2)/(2*2*numel(IDs));
DICE.NucRuber_right=(sum(NucRuber_right)*2)/(2*2*numel(IDs));
DICE.TopBrainstem=(sum(TopBrainstem)*2)/(2*2*numel(IDs));

DICE.OutlineBrainstem_left=(sum(OutlineBrainstem_left)*2)/(2*2*numel(IDs));
```

```
DICE.OutlineBrainstem_right=(sum(OutlineBrainstem_right)*2)/(2*2*numel(IDs)
);
DICE.LC_left=(sum(LC_left)*2)/(2*2*numel(IDs));
DICE.LC_right=(sum(LC_right)*2)/(2*2*numel(IDs));

DICE.BottomBrainstem=(sum(BottomBrainstem)*2)/(2*2*numel(IDs));
```

3.5. Calculate the in-plane distance between the slice-wise centroids of a template LC mask and single-subject transformed LC segmentations (a MATLAB script)

```
%% calculate in-plane distances: LC segmentations
clc;clear;close all
warning off

% paths
path_transformed = '/path/to/the/transformed/LCsegmentations/';
IDs=[]; % subject IDs

% load images
% aggregated LC segmentations, transformed & binarized
LC_tf_bin = spm_read_vols(spm_vol([path_transformed
'LCmask_heatmap_binary.nii'])); LC_tf_bin(find(LC_tf_bin==0))=NaN;
% template MNI LC mask
MNImask =
spm_read_vols(spm_vol(['/path/to/the/template/LCMask/mni_icbm152/mni_icbm15
2_LCmetaMask_MNI05_s01f_plus50_bin.nii'])); MNImask(MNImask==0)=0;

% identify coordinates of each voxel
[x_MNI,y_MNI,z_MNI] = ind2sub(size(MNImask),find(MNImask~=0));
[x_tf,y_tf,z_tf] = ind2sub(size(LC_tf_bin),find(~isnan(LC_tf_bin)));

% coordinates in double
positions_MNI = [x_MNI,y_MNI,z_MNI];
[~,idx1] = sort(positions_MNI(:,2));
slices_MNI_mask = unique(positions_MNI(:,3)); clear idx1


%% draw and check the transformed & aggregated LC segmentations in 3D space

close all

S1 = repmat([170],numel(x_tf),1);
S2 = repmat([200],numel(x_MNI),1);

hFig = figure();
axh = axes('Parent', hFig);
set(gca,'FontSize',25); hold on
hold(axh, 'all');
h2 = scatter3(x_MNI,y_MNI,z_MNI,S2,'d',...
    'MarkerEdgeColor','g',...
    'MarkerFaceColor',[0.4660 0.6740 0.1880]);
h1 = scatter3(x_tf,y_tf,z_tf,S1,'o',...
    'MarkerEdgeColor','r',...
    'MarkerFaceColor',[1 0.2 0.2],'MarkerFaceAlpha',.5);
xlabel('X','FontWeight','bold')
ylabel('Y','FontWeight','bold')
zlabel('Z','FontWeight','bold')
xlim([88 107])
ylim([91 99])
zlim([46 64])
view(axh, -33, 22);
grid(axh, 'on');
set(gca,'FontSize',18);
legend(axh, [h1,h2], {'Transformed Masks', 'MNI Meta LC template'});

%% find centroid voxel
```

```
% left LC
dummybase = zeros(193,229,193); % make a blank canvas

tmp_MNI=[x_MNI,y_MNI,z_MNI]; % tidy up the xyz coordinates from the
original mask

indL = tmp_MNI(:,1)<mean(x_MNI); % identify coordinates of left and right
coordsL = tmp_MNI(indL,:,:);
indR = tmp_MNI(:,1)>mean(x_MNI);
coordsR = tmp_MNI(indR,:,:);

leftLC=dummybase;
leftLC(sub2ind(size(leftLC),coordsL(:,1),coordsL(:,2),coordsL(:,3))) = 1;
rightLC=dummybase;
rightLC(sub2ind(size(leftLC),coordsR(:,1),coordsR(:,2),coordsR(:,3))) = 1;

[x_Left,y_Left,z_Left]=ind2sub(size(leftLC),find(leftLC~=0));
[x_Right,y_Right,z_Right]=ind2sub(size(rightLC),find(rightLC~=0));

centroid_L_mni =
[round(mean(x_Left)),round(mean(y_Left)),round(mean(z_Left))];
centroid_R_mni =
[round(mean(x_Right)),round(mean(y_Right)),round(mean(z_Right))];
centroid_both_mni =
[round(mean(x_MNI)),round(mean(y_MNI)),round(mean(z_MNI))];

%% extract centroid of single-subject masks

centroid_L = []; centroid_R =[];
for id = 1:length(IDs)

    clear tmp leftLC rightLC indL indR coordsL coordsR xi_Left yi_Left
zi_Left ...
        xi_Right yi_Right zi_Right

    LCsegs_individual_binary{id,1} =
spm_read_vols(spm_vol([path_transformed 'threshold025_' IDs{id}
'_conjmask_mni.nii']));

    dummybase = zeros(193,229,193); % make a blank canvas

    [xi,yi,zi] =
ind2sub(size(LCsegs_individual_binary{id,1}),find(LCsegs_individual_binary{
id,1}~=0));
    tmp=[xi,yi,zi]; % tidy up the xyz coordinates from the original mask

    indR = tmp(:,1)>97;%mean(xi); % identify coordinates of
    coordsR = tmp(indR,:,:);
    indL = tmp(:,1)<97;%mean(xi);
    coordsL = tmp(indL,:,:);

    leftLC=dummybase;
    leftLC(sub2ind(size(leftLC),coordsL(:,1),coordsL(:,2),coordsL(:,3))) =
1;

    rightLC=dummybase;
    rightLC(sub2ind(size(leftLC),coordsR(:,1),coordsR(:,2),coordsR(:,3))) =
1;

    % tmp_keren=kerenmask;%[x_keren,y_keren,z_keren];
```

```
    % leftLC = tmp_keren;leftLC(x_keren<mean(x_keren),:,:)=0; % choose
everything that's on the left
    % rightLC = tmp_keren;rightLC(x_keren>mean(x_keren),:,:)=0; % choose
everything that's on the right

    [xi_Left,yi_Left,zi_Left]=ind2sub(size(leftLC),find(leftLC~=0));
    left_indivs{id,1} = [xi_Left,yi_Left,zi_Left];
    [xi_Right,yi_Right,zi_Right]=ind2sub(size(rightLC),find(rightLC~=0));
    right_indivs{id,1} = [xi_Right,yi_Right,zi_Right];

    centroid_L(id,:) =
[round(mean(xi_Left)),round(mean(yi_Left)),round(mean(zi_Left))];
    centroid_R(id,:) =
[round(mean(xi_Right)),round(mean(yi_Right)),round(mean(zi_Right))];
    centroid_both(id,:) =
[round(mean(xi)),round(mean(yi)),round(mean(zi))];
end



%% distance from the centroid points, slicewise analysis

% find centre of each slice
MNI_left = [x_Left,y_Left,z_Left];
MNI_right = [x_Right,y_Right,z_Right];
MNI_slices_L = unique(z_Left);
MNI_slices_R = unique(z_Right);

for slices = 1:length(MNI_slices_L)
    clear tmp zind
    % sort the left
    zind=z_Left==MNI_slices_L(slices);
    tmp = MNI_left(zind',:,:);
    if size(MNI_left(zind',:,:),1)==1
    Lcentroids_mni(slices,:) = [tmp(:,1:2) MNI_slices_L(slices)];
    else
    Lcentroids_mni(slices,:) = [mean(tmp(:,1:2)) MNI_slices_L(slices)];
    end
end

for slices = 1:length(MNI_slices_R)

    clear tmp zind
    % sort the left
    zind=z_Right==MNI_slices_R(slices);
    tmp = MNI_right(zind',:,:);
    if size(MNI_right(zind',:,:),1)==1
    Rcentroids_mni(slices,:) = [tmp(:,1:2) MNI_slices_R(slices)];
    else
    Rcentroids_mni(slices,:) = [mean(tmp(:,1:2)) MNI_slices_R(slices)];
    end

end


% now find centres of individual masks
Lcentroids_indiv=[];Rcentroids_indiv=[];
for id=1:length(IDs)

    clear maskL maskR slicesL slicesR sl
```

```
    % first, left
    maskL=right_indivs{id,1}; % there's something wrong with this left and
right
    slicesL=unique(maskL(:,3));
    for sl=1:length(slicesL)
        clear tmp zind
        % sort the left
        zind=maskL(:,3)==slicesL(sl);
        tmp = maskL(zind',:,:);
        if size(tmp,1)== 1
            Lcentroids_indiv{id,1}(sl,:) = [tmp(:,1:2) slicesL(sl)];
        else
            Lcentroids_indiv{id,1}(sl,:) = [mean(tmp(:,1:2)) slicesL(sl)];
        end
    end


    % then, right
    maskR=left_indivs{id,1}; % there's something wrong with this left and
right
    slicesR=unique(maskR(:,3));
    for sl=1:length(slicesR)
        clear tmp zind
        % sort the left
        zind=maskR(:,3)==slicesR(sl);
        tmp = maskR(zind',:,:);
        if size(tmp,1)== 1
            Rcentroids_indiv{id,1}(sl,:) = [tmp(:,1:2) slicesR(sl)];
        else
            Rcentroids_indiv{id,1}(sl,:) = [mean(tmp(:,1:2)) slicesR(sl)];
        end
    end

end

%% calculate distance of centroids from the template centroids, slicewise

ED_slice_L=[]; ED_slice_R=[];
for id=1:length(IDs)

    % find the slices that matches
    clear Lslc_tmp A B
    Lslc_tmp = Lcentroids_indiv{id,1}(:,3);
    indslc1=ismember(MNI_slices_L,Lslc_tmp,'rows');
    indslc2=ismember(Lslc_tmp,MNI_slices_L,'rows');

    A = Lcentroids_mni(indslc1,:);
    B = Lcentroids_indiv{id,1}(indslc2,:);
    for k=1:size(B,1)
        clear v
        v    = B(k,:) - A(k,:);
        ED_slice_L{id,1}(k) = sqrt(nansum(v .^ 2));
    end

    % find the slices that matches
    clear Rslc_tmp A B
    Rslc_tmp = Rcentroids_indiv{id,1}(:,3);
    indslc1=ismember(MNI_slices_R,Rslc_tmp,'rows');
    indslc2=ismember(Rslc_tmp, MNI_slices_R,'rows');
```

```
    A = Rcentroids_mni(indslc1,:);
    B = Rcentroids_indiv{id,1}(indslc2,:);

    for k=1:size(A,1)
        clear v
        v    = B(k,:) - A(k,:);
        ED_slice_R{id,1}(k) = sqrt(nansum(v .^ 2));
    end

end


% additional processing
ED_slice_mean_L = cellfun(@nanmean, ED_slice_L);
ED_slice_mean_R = cellfun(@nanmean, ED_slice_R);

%% simple stats

means(1,1)=nanmean(ED_slice_mean_L);
means(1,2)=nanmean(ED_slice_mean_R);
stds(1,1)=nanstd(ED_slice_mean_L);
stds(1,2)=nanstd(ED_slice_mean_R);
medians(1,1)=nanmedian(ED_slice_mean_L);
medians(1,2)=nanmedian(ED_slice_mean_R);

fprintf('\n left LC mean: %1.4f, STD: %1.4f, median: %1.4f \n right LC
mean: %1.4f, STD: %1.4f, median: %1.4f \n', ...
    means(1,1), stds(1,1), medians(1,1), means(1,2), stds(1,2),
medians(1,2))
```

4. References

Avants, B. B., Tustison, N. J., Song, G., Cook, P. A., Klein, A., Gee, J. C., 2011. A reproducible evaluation of ANTs similarity metric performance in brain image registration. NeuroImage 54 (3), 2033-2044.

Beckmann, C. F., Smith, S. M., 2004. Probabilistic independent component analysis for functional magnetic resonance imaging. IEEE Transactions on Medical Imaging 23 (2), 137-152.

Behzadi, Y., Restom, K., Liau, J., Liu, T. T., 2007. A component based noise correction method (CompCor) for BOLD and perfusion based fMRI. NeuroImage, 37(1), 90–101.

Dahl, M. J., Mather, M., Werkle-Bergner, M., Kennedy, B. L., Qiao, Y., Shi, Y., Ringman, J. M., 2022. Locus coeruleus integrity is related to tau burden and memory loss in autosomal-dominant Alzheimer's disease. Neurobiology of Aging 112, 39-54.

Fonov, V., Evans, A. C., Botteron, K., Almli. R., McKinstry, R. C., Collins, D. L., the Brian Development Cooperative Group, 2011. Unbiased average age-appropriate atlases for pediatric studies. NeuroImage 54 (1), 313-327.

Hämmerer, D., Callaghan, M.F., Hopkins, A., Kosciessa, J., Betts, M., Cardenas-Blanco, A., Kanowski, M., Weiskopf, N., Dayan, P., Dolan, R.J., Düzel, E., 2018. Locus coeruleus integrity in old age is selectively related to memories linked with salient negative events. Proceedings of the National Academy of Sciences 115 (9), 2228-2233.

Klein, A., Andersson, J., Ardekani, B. A., Ashburner, J., Avants, B., Chiang, M.-C., Christensen, G. E., Collins, D. L., Gee, J., Hellier, P., Song, J. H., Jenkinson, M., Lepage, C., Rueckert, D., Thompson, P., Vercauteren, T., Woods, R. P., Mann, J. J., Parsey, R. V., 2009. Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration. NeuroImage 46, 786-802.

Klein, A., Ghosh, S. S., Avants, B., Yeo, B. T. T., Fischl, B., Ardekani, B. A., Gee, J. C., Mann, J. J., Parsey, R. V., 2010. Evaluation of volume-based and surface-based brain image registration methods. NeuroImage 51, 214-220.

Sasaki, M., Shibata, E., Tohyama, K., Takahashi, J., Otsuka, K., Tsuchiya, K., Takahashi, S., Ehara, S., Terayama, Y., Sakai, A. 2006. Neuromelanin magnetic resonance imaging of locus ceruleus and substantia nigra in Parkinson's disease. Neuroreport, 17, 1215-1218.

Tustison, N. J., Avants, B. B., 2013. Explicit B-spline regularization in diffeomorphic image registration. Frontiers in Neuroinformatics 7(39), 1-13.