

Quality assessment of spatial transformation of locus coeruleus (LC) functional and structural magnetic resonance imaging (MRI) data

1. Introduction
 - 1.1. Spatial transformation procedure
 - 1.2. Overview of landmark assessment procedure for functional MRI data
 - 1.3. The background and the goal of this procedure
2. Protocol
 - 2.1. Preparation
 - 2.1.1. Required data
 - 2.1.2. Required software
 - 2.2. Procedure
 - 2.2.1. Selecting landmarks
 - 2.2.2. Drawing landmarks
 - 2.2.3. Aggregating single-subject landmarks for preliminary inspection
 - 2.2.4. Statistical quality assessment of aggregated landmarks of transformed functional images
 - 2.2.5. Statistical quality assessment of transformed structural images
 - 2.3. Examples of spatial transformation failures and fixes
 - 2.3.1. Suboptimal registration of mean functional images
 - 2.3.2. Suboptimal normalisation near the skull in mean functional images
3. Codes
 - 3.1. The spatial transformation pipeline (a shell script)
 - 3.2. Generate Nifti landmark overlay image (a MATLAB script)
 - 3.3. Generate video of transformed mean functional or structural images (a MATLAB script)
 - 3.4. Calculate the Euclidean distance of single-subject landmarks (a MATLAB script)
 - 3.5. Calculate the Euclidean distance between the slice-wise centroids of a template LC mask and single-subject transformed LC segmentations (a MATLAB script)
4. References

1. Introduction

1.1. Spatial transformation procedure

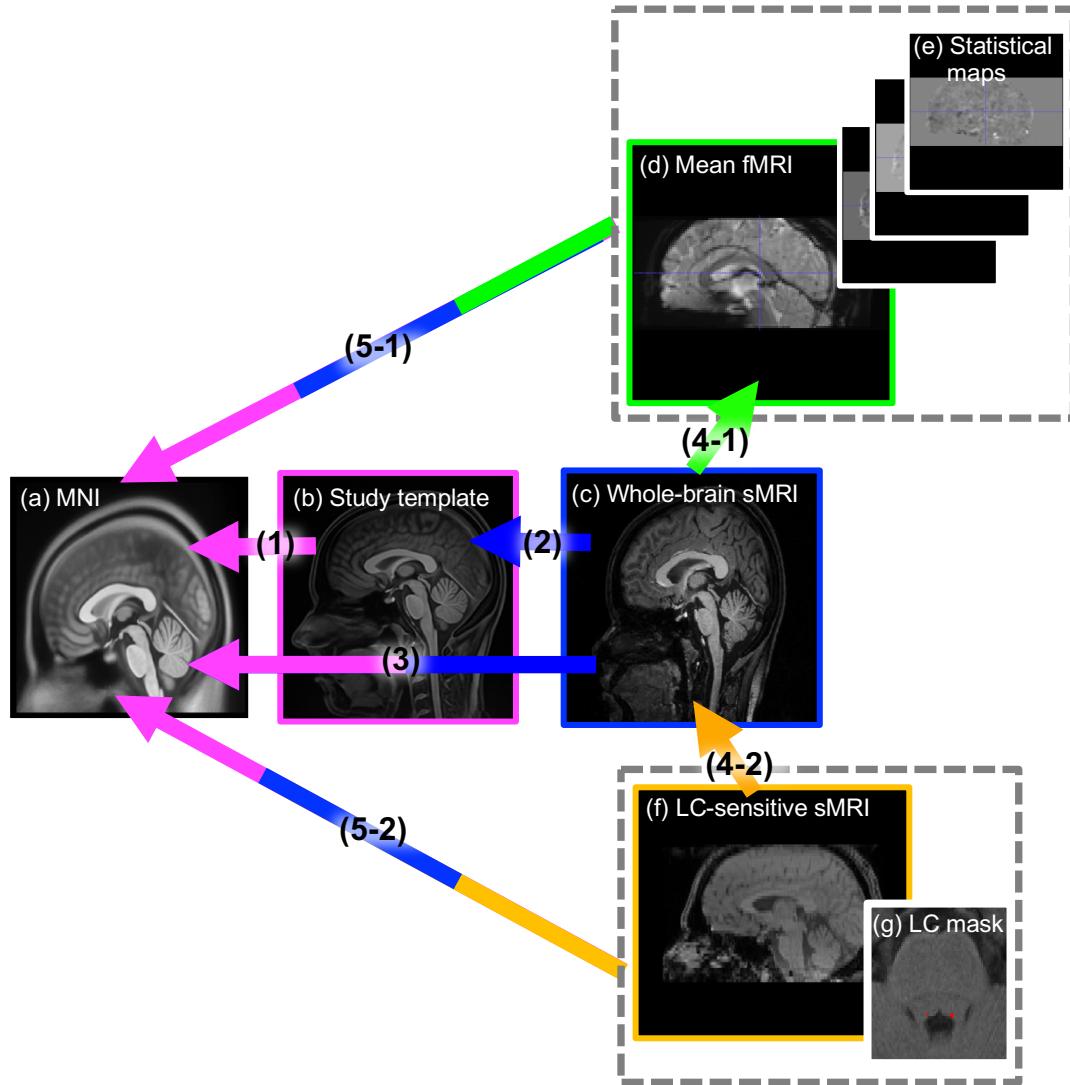


Figure 1. A schematic of the spatial transformation procedure

Previous to the landmark-based spatial transformation quality assessment, pre-processing of the images in order to prepare for the spatial transformation was done. Then, a series of transformations were executed on the mean functional images (**d**), whole-brain structural images (**c**), neuromelanin-sensitive structural images (**f**), and LC segmentations (**g**). The below is a modified excerpt from the main text describing the pre-processing and the spatial transformation steps described in the figure above to help the readers to understand the background of the quality assessment procedure.

"For each participant, two-session functional images were first slice-time corrected, unwarped, and realigned using Statistical Parametric Mapping (SPM12) in the MATLAB environment (Version 2015a) using default parameters, except the 6th degree interpolation option for both estimating and reslicing. This generated a mean functional image per person used for spatial transformation of the structural and functional images (cf. **Figure 1d**). The higher-degree interpolation method for realignment was selected to minimise the spatial noise between the two concatenated functional scan sessions. Thereafter, the time-series functional images were smoothed with SPM using a $3 \times 3 \times 3$ mm kernel in the native space before running single-subject general linear models (GLM) for estimated task-related contrasts in SPM. As physiological noise parameters were not recorded during data collection, they were retroactively corrected using a component-based method (CompCor) during the single-subject GLM analyses (Behzadi et al., 2007). The GLM analysis generated a set of statistical contrast maps in the native space per subject (**e**) that was ready for transformation into the MNI space.

Individual T1-weighted whole-brain structural images were bias-corrected using Advanced Normalisation Tools' *N4BiasFieldCorrection* function (**c**) to correct for field-related inhomogeneity (ANTs, Version 2.3.1). Finally, a study-specific template (**b**) was created from these bias field-corrected structural whole-brain images using the *antsMultivariateTemplateConstruction2* function of ANTs.

The LC was manually segmented on the neuromelanin-sensitive images (c.f. **Figure 1f** and **1g**) using ITK-Snap software (version 3.6.0-RC1) by two independent expert raters. Final LC segmentations contain only the overlapping voxels from both raters (cf. **Figure 1g**) (Sørensen–Dice coefficient=0.60±0.17; see also Häammerer et al. 2018 for more details on LC segmentation generation).

...

To prepare the transition from the native to the group or MNI space, the study-specific group template (**b**) was non-linearly registered to the MNI space using *antsRegistrationSyn* (**step 1**). Subsequently, the individual whole-brain structural MPRAGE image (**c**) in the native space was registered non-linearly to the template (**b**) using *antsRegistrationSyn* (**step 2**) and then transformed into the MNI space (**a**) with the concatenated transformation matrix and deformation fields generated from **steps (1)** and **(2)** using *antsApplyTransforms* (**step 3**). Then, individual mean functional images (**d**), LC-sensitive structural images (**f**), and the LC segmentations (**g**) in the native space were rigidly registered to the individual structural whole-brain image (**c**) using *antsRegistrationSyN* (**steps 4-1 and 4-2, respectively**). With the concatenated transformation matrices and deformation fields acquired from steps **(1)**, **(2)**, and **(4-1)**, the mean functional images (**d**) and individual contrast images (**e**), which are in the same space as individual mean functional images, were transformed to the MNI space (**a**) non-linearly in one transformation step using *antsApplyTransforms* (**step 5-1**). Similarly,

concatenated transformation matrices and deformation fields acquired from steps **(1)**, **(2)**, and **(4-2)** were used to transform LC segmentations **(g)** delineated on neuromelanin-sensitive structural images **(f)** non-linearly into the MNI space **(step 5-2)** using *antsApplyTransforms*. All nonlinear spatial transformations were implemented at the 4th degree B-spline interpolation. The MNI-transformed individual LC segmentations were then binarized at 0.25 threshold using the *mri_binarize* function of FreeSurfer.”

1.2. Overview of landmark assessment procedure

After all subject datasets have been processed, the quality of the transformation was examined. This examination was done by selecting the landmarks in the MNI space, drawing them on the individual transformed mean functional images, and calculating Euclidean distances between the MNI-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 to quickly identify abnormal registration. More detailed procedures will be explained in the walkthrough section.

1.3. The background and the goal of this procedure

The LC is so small that even a slight spatial deviation of a voxel size could significantly reduce the statistical power of group-level analyses. For more robust functional assessment of such structure, it is not only important to register and normalise each single subject functional image properly onto the group space, but also to control the quality of the transformed images and rectify any problems that arose from the spatial transformation. With the quality assessment protocol that are introduced in detail below, we aim to achieve more reliable and powerful voxel-wise analyses of the functional LC imaging data than existing approaches in the field.

2. Protocol

2.1. Preparation

2.1.1. Required data

The files that are needed for the assessment are:

- the MNI space to which all images are transformed
- transformed mean functional images of each subject

The MNI space can also be a group space depending on the analysis that will be done with the transformed functional data. In the example dataset, an MNI space 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 4.

2.2. Procedure

2.2.1. Selecting landmarks

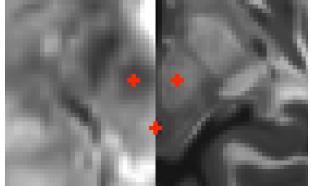
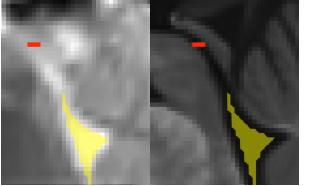
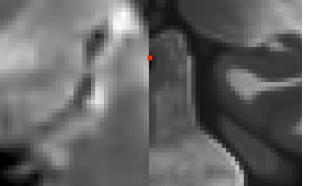
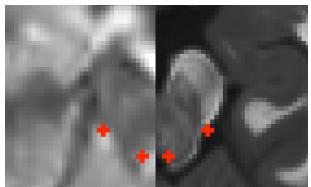
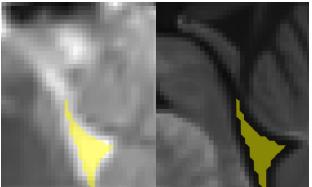
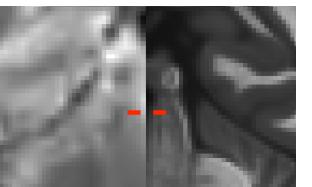
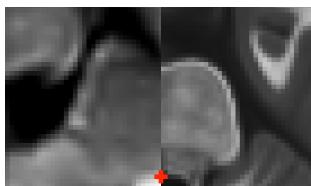
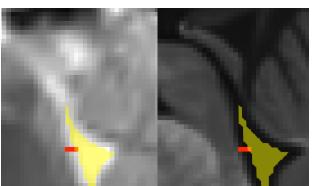
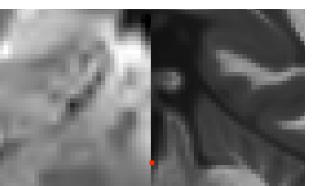
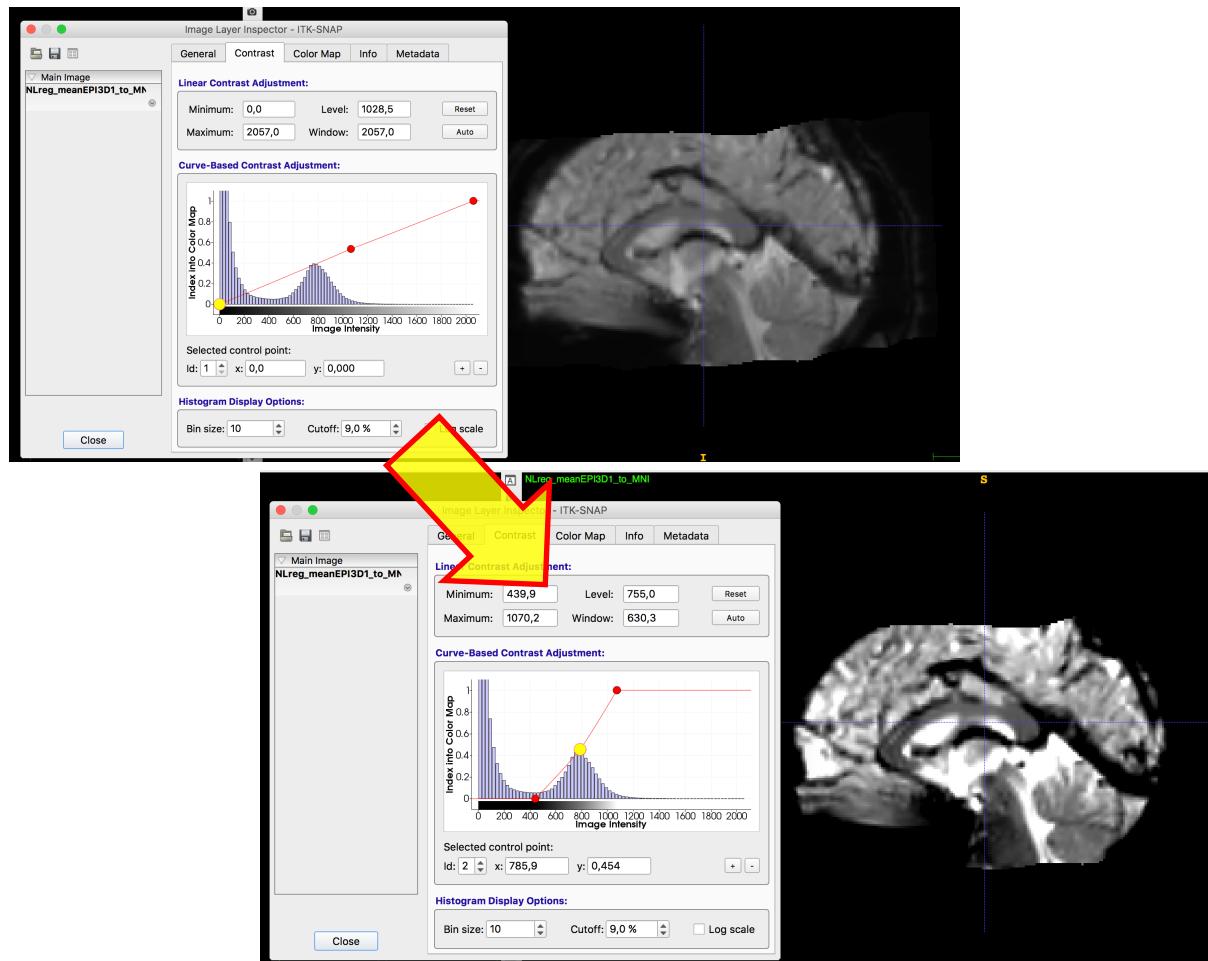
	Axial view	Sagittal view	Coronal view
Top of the brainstem and Nuclei Ruber (slice 71)			
LC and the brainstem outline (slice 60)			
Lower posterior border of brainstem (slice 50)			

Figure 2. Selecting landmarks for functional data

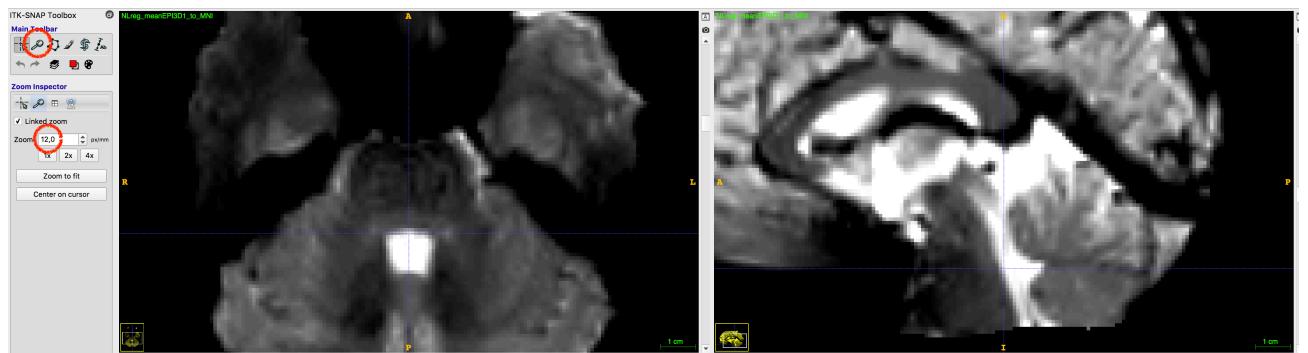
Selecting an appropriate set of landmarks depends on the region of interest (ROI). In this manual's ROI, LC, is located in the brainstem near the lateral floor of the fourth ventricle. Therefore, selecting landmarks in the upper brainstem that delineate the bounds of LC location is most helpful for the 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. Based on these criteria, the landmarks specified in the table above were selected in the main text. The top of the brainstem and nuclei Ruber were selected as a landmark to indicate the upper bound of the larger ROI area of the study, the brainstem. Also, Nucleus Ruber is distinctly visible in the functional images, which makes it an ideal candidate for a landmark. The LC and the outline of the brainstem was selected to delineate the bounds of LC activation. Lastly, the lower posterior border of brainstem was selected to indicate the lowest bound of possible LC

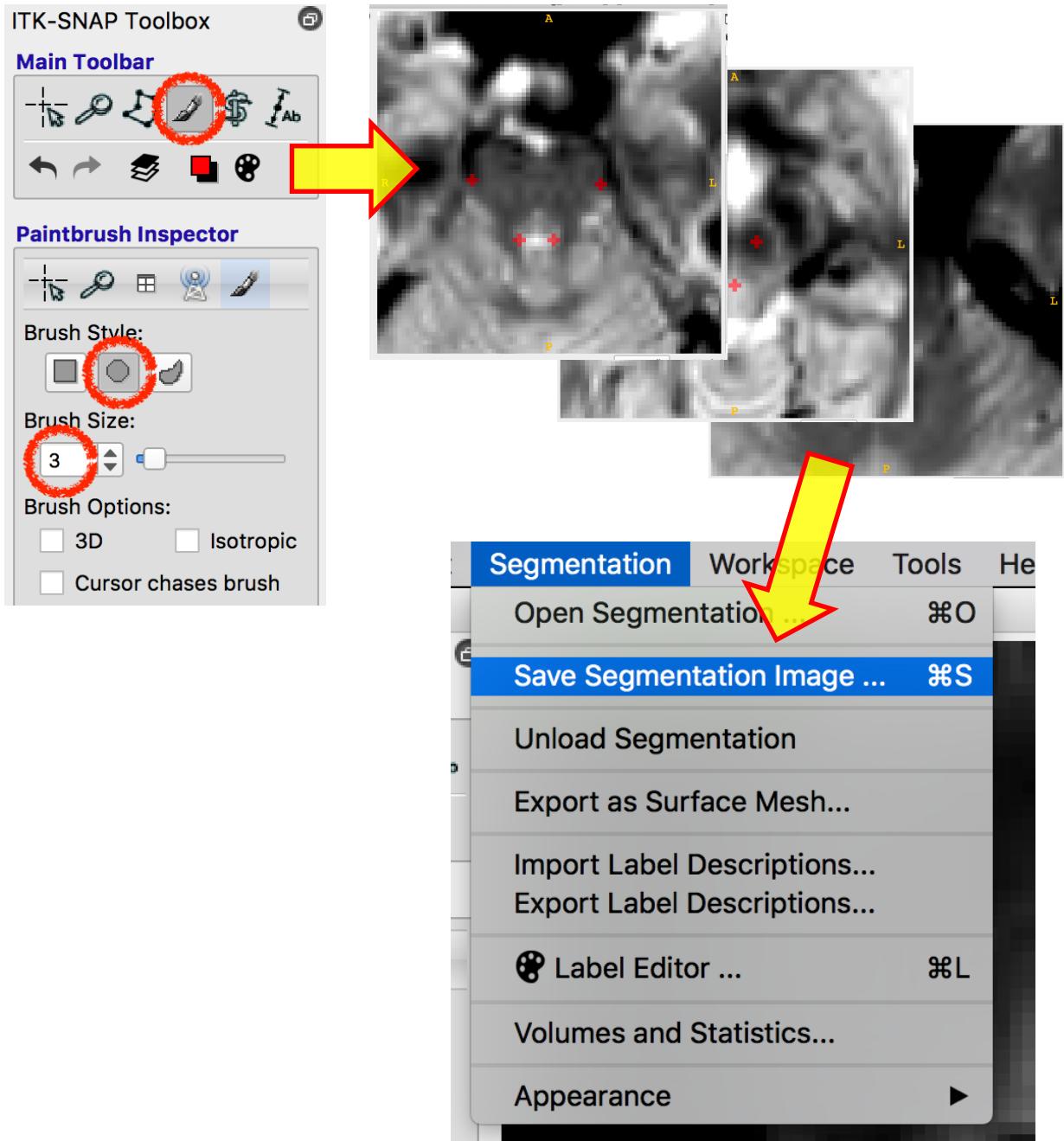
location. Once the landmarks are selected, evaluators should familiarise themselves with them.



2.2.2. Drawing landmarks

Next, evaluators inspect one of the transformed single-subject mean functional images. If necessary, adjust the contrast slider so that the fourth ventricle is as bright and distinguished as it can be. After adjusting the contrast of the image, magnify the image that you can draw the landmarks more precisely.





Then, using the size-3 round brush segmentation tool, place the landmarks on the transformed mean functional images in axial view. However, use the anatomical clues from other views to aid the landmark placement. For example, when drawing 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 the image is overlaid, adjust the transparency setting to examine whether the landmark segmentations



are in the right place. Take note of the subject ID if the landmarks were badly placed. 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 executing the custom code from section 4 of this manual. By performing this step together with the frame-by-frame video of transformed images (the code to generate the video is in section 3-c, an example of this video is in the section 2.4 of the supplementary material), outliers are easily identified.

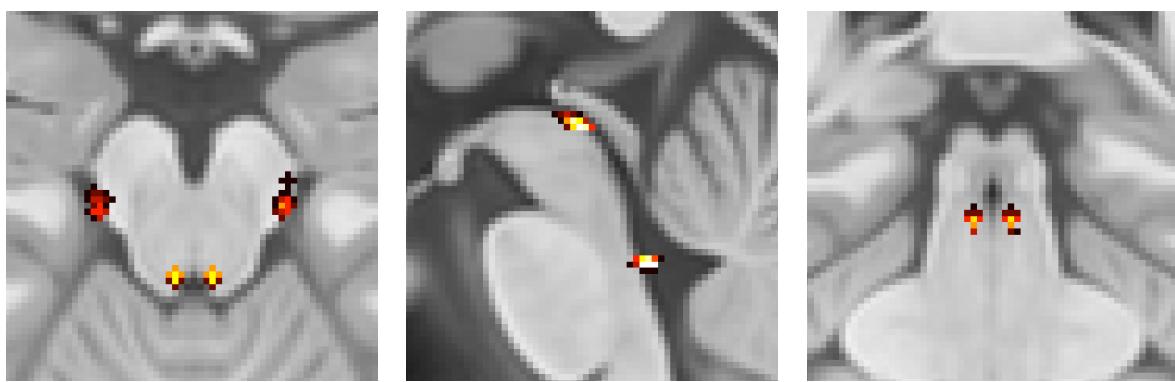


Figure 3. A heatmap of aggregated landmarks overlaid on the MNI space

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

Once all faulty transformation issues are addressed and rectified, statistical analyses of the quality assessment is performed to quantify the transformation precision. An example code for the analyses is in the section 3.d. Through this step, the spatial transformation can be further refined. The detailed background and example results are described in the main text.

2.2.5. Statistical quality assessment of transformed structural images

Additionally, 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.a of this manual, the LC segmentations are transformed onto the MNI space using the transformation matrices and the deformation fields generated from warping neuromelanin-sensitive structural image (f) to the MNI space (step 5-2). The quantification of the assessment can be done by calculating Euclidean distance between the slice-wise centroids of the template LC mask in the MNI space, e.g. the meta LC mask by Dahl et al.

(2021) in this manual's case, and the transformed LC segmentations of each subject. An example code for the calculating the distance is in the section 3.e.

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 the errors are described, should such errors arise in a pipeline that is tailored to a large dataset and pruned for efficiency.

2.3.1. Suboptimal registration of mean functional images

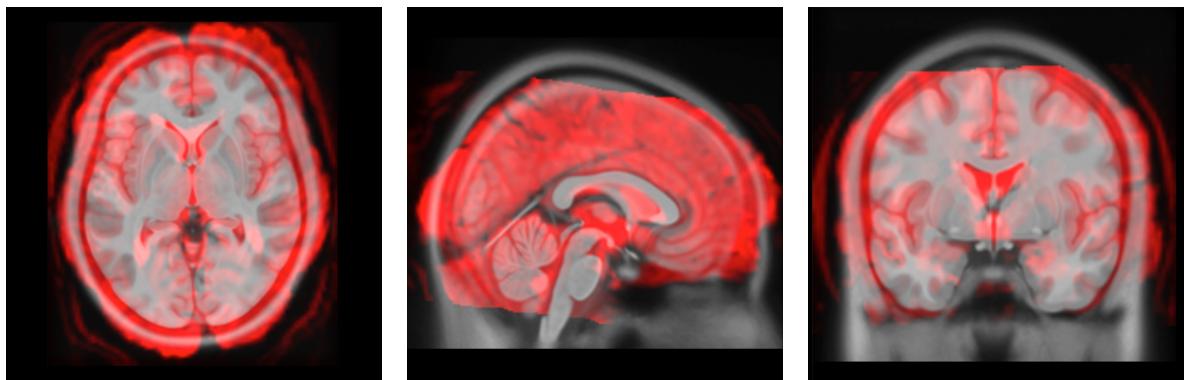


Figure 4. An example of suboptimal registration in the transformed mean functional image. The transformed mean functional image is indicated as red additive overlays on the MNI template image in each view.

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 is seen the skull of the structural image is fitted to the tissue of the mean functional image.

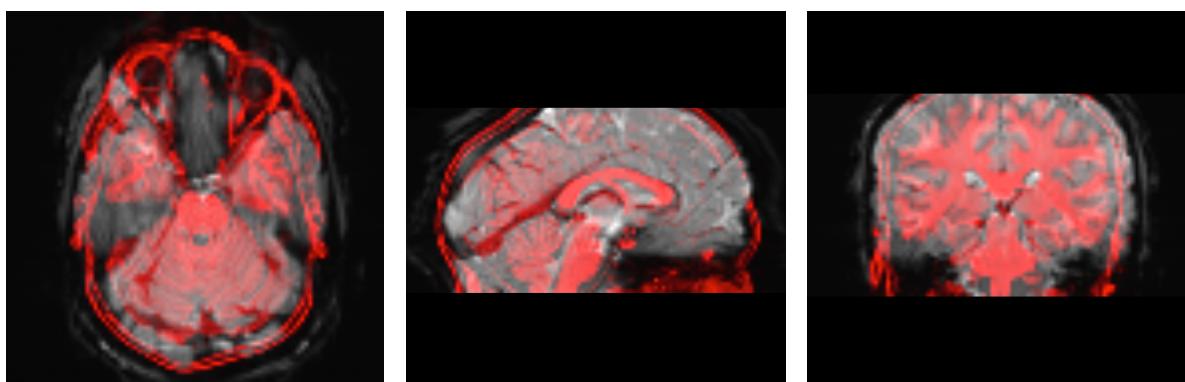


Figure 5. Rigid registration failure of the structural whole-brain image to the mean functional image. The structural image was registered to the native mean functional image and is indicated as red additive overlays on the mean functional image.

This issue seems to occur because the *antsRegistrationSyN* algorithm 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) with **-x** switch.

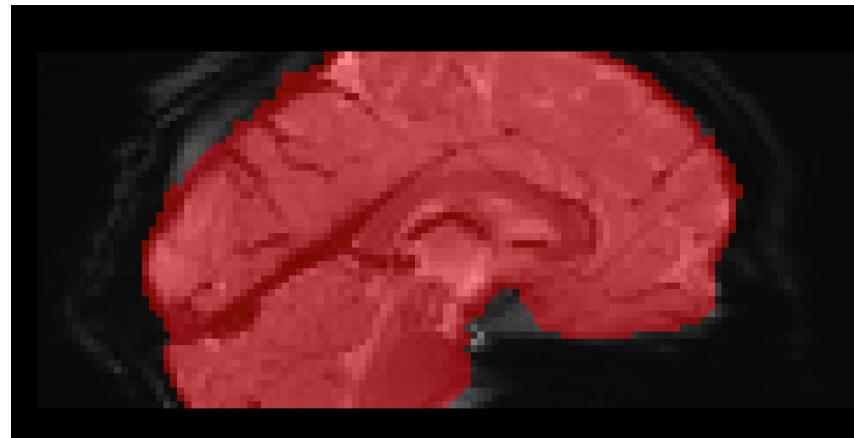


Figure 6. An example brain-only mean functional image mask overlaid on the mean functional image. The transparent red image indicates the area where the mask covers.

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.

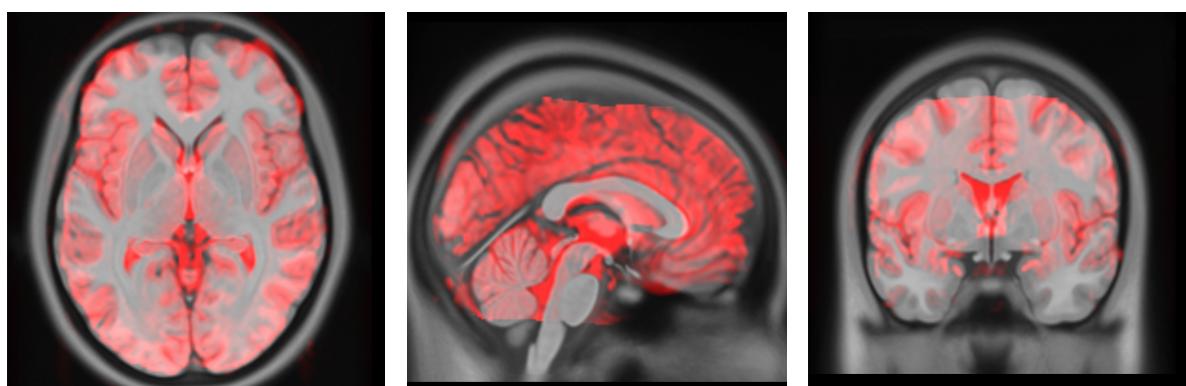


Figure 7. An example of successful spatial transformation of the mean functional image after employing the brain-only mask in the step 4-1 of Figure 1. The transformed mean functional image is indicated as red additive overlays on the MNI template in each view.

2.3.2. Suboptimal normalisation near the skull in mean functional images

Another example of transformation failure is bad 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 above.

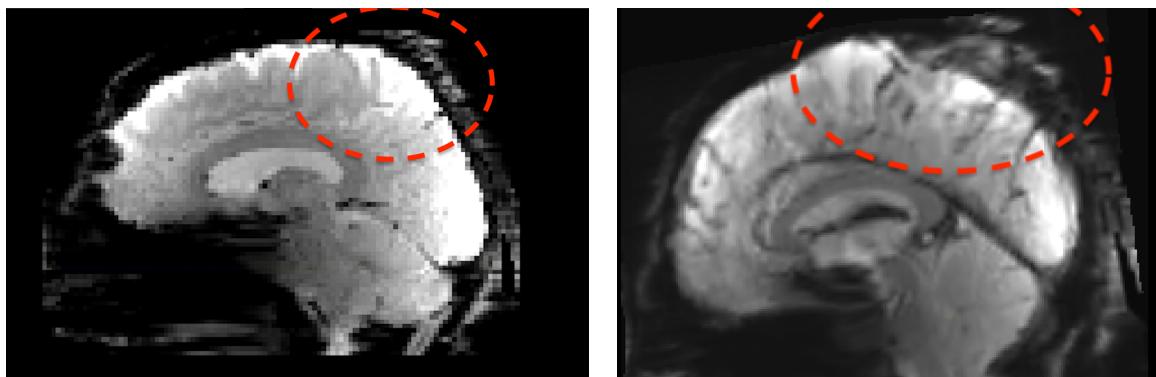


Figure 8. An example of suboptimal normalisation in the mean functional images. The left pane shows the native mean functional image, and the right pane shows the transformed mean functional image.

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.

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/yyi/temp/ED_coreg/"${ID}"/
MNI=/mnt/work/yyi/temp/ED_coreg/mni_icbm152_t1_tal_nlin_asym_09c.nii
template=/mnt/work/yyi/temp/ED_coreg/pilot_template.nii.gz

# LC segmentation
mask=$(ls -t "${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 -> MNI
antsApplyTransforms -d 3 -v 1 -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

# 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 1 -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 1 -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
"${folder}"/data/coreg_t1slab_to_T1mean_0GenericAffine.mat -i "${mask}" -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 1 -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 Nifti landmark overlay image (a MATLAB script)

```
%% Check the coregistration with EPI landmarks
% make a plot
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)

ids = [];% put your ID list

nid = length(ids);

%% make

cc = 0; % reset the counter
averplot = zeros(193,229,193); % pre-assign the dummy variable for plotting
the masks
figure; % open sketchbook
for i = 1:length(ids)

    averplotdum = zeros(193,229,193);
    name = num2str(ids(i));
    data = spm_read_vols(spm_vol([path_root name '/data/LCmask_check_' name
'.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 LC masks
    ycoordz = y(find(data(:) == 1)); uycoordz = unique(ycoordz);
    ylevel = floor(median(uycoordz)); % take middle one

    xcoordz = x(find(data(:) == 1)); uxcoordz = unique(xcoordz);
    xlevel = ceil(median(uxcoordz)); % take middle one
```

```

zcoordz = z(find(data(:) == 1)); uzcoordz = unique(zcoordz);
zlevel = ceil(median(uzcoordz)); % take middle one

% right now, z position, which is an axial view, seems to be the most
% appropriate slice to check the coregistration. if you want to take a
% look at the other views, please change the script accordingly
dumz = squeeze(data(:,:,:,zlevel));
imagesc(dumz);title([name ' ' num2str(i) ]);% middle slice
cc = cc+1;
zstore(cc) = zlevel;
averplotdum(:,:,:,:,zlevel) = dumz; % average points and squeeze them in

%
dumy = squeeze(data(:,:,ylevel,:));
imagesc(dumy);title([name ' ' num2str(i) ]);% middle slice
cc = cc+1;
ystore(cc) = ylevel;
averplotdum(:,:,ylevel,:)= dumy;

%
dumx = squeeze(sum(data(:,(xlevel-1):(xlevel+1),:),2));
imagesc(dumx);title([name ' ' num2str(i) ]);% middle slice
cc = cc+1;
xstore(cc) = xlevel;
averplotdum(xlevel,:,:,:)= dumx;

averplot = averplot + averplotdum; % record the points
imagesc(squeeze(averplotdum(:,:,:,:,zlevel)));hold on % and plot

%
imagesc(squeeze(averplotdum(xlevel,:,:,:)));hold on % x position plot
imagesc(squeeze(averplotdum(:,:,ylevel,:)));hold on % y position plot

clear dumx dumy dumz

end

averplot = averplot/cc; % get average

%% generate the overlay image in 3D space

% one mask where all voxel > 0 are 1

```

```
averplot_nifti = averplot; averplot_nifti(find(averplot_nifti(:)>0)) = 1;
hdr = spm_vol([path_root '/data/landmarks.nii']); % pick just any header from
a file
hdr.fname = [path_save 'landmark_heatmap.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 coregistered images
% make movie of epis to see whetehr OK aligned

%% 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
'/data/transformed_image.nii']));
    
    % position the slice - dimensions are: (x=sagittal, y=coronal,
z=axial)
    if strcmpi(vidview,'sagittal')
        dum = squeeze(data(:,:, :, :)); % sagittal
    elseif strcmpi(vidview,'coronal')
        dum = squeeze(data(:, :, :, :)); % coronal view
    elseif strcmpi(vidview,'axial')
        dum = squeeze(data(:, :, :, :)); % 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 Euclidean distance of single-subject landmarks (a MATLAB script)

```
%% LANDMARK PLOTTING: create 3D plots of created heatmaps, and compare

clc;clear;close all

% set env
path_transformed = '/path/to/your/individual/landmarks/';
load('/path/to/your/subject/IDs/IDs.mat') % load IDs

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

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

% write down the coordinates of MNI landmarks
MNI.NucRuber=[];
MNI.NucRuber.left=[102 113 71];
MNI.NucRuber.right=[92 113 71];

MNI.TopBrainstem=[97 102 71];

MNI.OutlineBrainstem=[];
MNI.OutlineBrainstem.left=[109 103 60];
MNI.OutlineBrainstem.right=[85 103 60];

MNI.LC=[];
MNI.LC.left=[100 97 60];
MNI.LC.right=[94 97 60];

MNI.BottomBrainstem=[97 94 50];

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 positionX_sorted cluster1 cluster2 cluster3

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

% ruber right
cluster1 = min(positionX_sorted(:,1));
cluster_med1 =
positionX_sorted(find(positionX_sorted(:,1)==cluster1+1),:);
cluster_medpt{subj,1} = cluster_med1( find( cluster_med1(:,2)==ceil(
median(cluster_med1(:,2))) ),: );

% top curve
cluster2 = min(positionX_sorted(positionX_sorted(:,1)>cluster1+2,1));
cluster_med2 =
positionX_sorted(find(positionX_sorted(:,1)==cluster2+1),:);
cluster_medpt{subj,2} = cluster_med2( find( cluster_med2(:,2)==ceil(
median(cluster_med2(:,2))) ),: );

% ruber left
cluster3 = min(positionX_sorted(positionX_sorted(:,1)>cluster2+2,1));
cluster_med3 =
positionX_sorted(find(positionX_sorted(:,1)==cluster3+1),:);
cluster_medpt{subj,3} = cluster_med3( find( cluster_med3(:,2)==ceil(
median(cluster_med3(:,2))) ),: );

transformed_landmark_coords{subj,1}.TopSlice = [];
transformed_landmark_coords{subj,1}.TopSlice.NucRuber_left =
cluster_medpt{subj,3};
transformed_landmark_coords{subj,1}.TopSlice.NucRuber_right =
cluster_medpt{subj,1};
transformed_landmark_coords{subj,1}.TopSlice.TopBrainstem =
cluster_medpt{subj,2};
clear cluster_medpt

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

clear idx3 positionY positionX_sorted cluster1 cluster2 cluster3

positionSlice = tmp_sorted(find(tmp_sorted(:,3)==slices(2)),:);
[~,idx3] = sort(positionSlice(:,1)); positionX_sorted =
positionSlice(idx3,:);

% brainstem outline right
cluster1 = min(positionX_sorted(:,1));
cluster_med1 =
positionX_sorted(find(positionX_sorted(:,1)==cluster1+1),:);
cluster_medpt{subj,1} = cluster_med1( find( cluster_med1(:,2)==ceil(
median(cluster_med1(:,2))) ),: );

% LC right
cluster2 = min(positionX_sorted(positionX_sorted(:,1)>(cluster1+2),1));
cluster_med2 =
positionX_sorted(find(positionX_sorted(:,1)==cluster2+1),:);

```

```

    cluster_medpt{subj,2} = cluster_med2( find( cluster_med2(:,2)==ceil(
median(cluster_med2(:,2))) ),:, );

% LC left
cluster3 = min(positionX_sorted(positionX_sorted(:,1)>cluster2+2,1));
cluster_med3 =
positionX_sorted(find(positionX_sorted(:,1)==cluster3+1),:);
cluster_medpt{subj,3} = cluster_med3( find( cluster_med3(:,2)==ceil(
median(cluster_med3(:,2))) ),:, );

% brainstem outline left
cluster4 = min(positionX_sorted(positionX_sorted(:,1)>cluster3+2,1));
cluster_med4 =
positionX_sorted(find(positionX_sorted(:,1)==cluster4+1),:);
cluster_medpt{subj,4} = cluster_med4( find( cluster_med4(:,2)==ceil(
median(cluster_med4(:,2))) ),:, );

transformed_landmark_coords{subj,1}.MidSlice = [];
transformed_landmark_coords{subj,1}.MidSlice.OutlineBrainstem_left =
cluster_medpt{subj,4};
transformed_landmark_coords{subj,1}.MidSlice.OutlineBrainstem_right =
cluster_medpt{subj,1};
transformed_landmark_coords{subj,1}.MidSlice.LC_left =
cluster_medpt{subj,3};
transformed_landmark_coords{subj,1}.MidSlice.LC_right =
cluster_medpt{subj,2};
clear cluster_medpt

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

clear idx3 positionY positionX_sorted cluster1 cluster2 cluster3
cluster4

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

% brainstem bottom
cluster1 = min(positionX_sorted(:,1));
cluster_med1 =
positionX_sorted(find(positionX_sorted(:,1)==cluster1+1),:);
cluster_medpt{subj,1} = cluster_med1( find( cluster_med1(:,2)==ceil(
median(cluster_med1(:,2))) ),:, );

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

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

end;clear subj

disp('coordinates collected')

%% calculate euclidian distances in single subject level

```

```

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

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

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

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

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

    EuclideanDistances_indv{subj,1}.OutlineBrainstem.left=sum((transformed_landmark_coords{subj,1}.MidSlice.OutletBrainstem_left-
MNI.OutletBrainstem.left).^2).^0.5;

    EuclideanDistances_indv{subj,1}.OutlineBrainstem.right=sum((transformed_landmark_coords{subj,1}.MidSlice.OutletBrainstem_right-
MNI.OutletBrainstem.right).^2).^0.5;

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

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

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

    EuclideanDistances_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)=EuclideanDistances_indv{subj,1}.NucRuber.left;
NucRuber_right=subj,1)=EuclideanDistances_indv{subj,1}.NucRuber.right;
TopBrainstem(subj,1)=EuclideanDistances_indv{subj,1}.TopBrainstem;

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

OutlineBrainstem_right=subj,1)=EuclideanDistances_indv{subj,1}.OutlineBrainstem.right;
LC_left=subj,1)=EuclideanDistances_indv{subj,1}.LC.left;
LC_right=subj,1)=EuclideanDistances_indv{subj,1}.LC.right;

BottomBrainstem(subj,1)=EuclideanDistances_indv{subj,1}.BottomBrainstem;

end

```

```
spssID = cell2mat(cellfun(@str2num, IDs, 'UniformOutput',false));  
  
T=table(NucRuber_left,NucRuber_right,TopBrainstem,OutlineBrainstem_left,OutlineBrainstem_right,...  
    LC_left,LC_right,BottomBrainstem);  
ED_export  
=[spssID',NucRuber_left,NucRuber_right,TopBrainstem,OutlineBrainstem_left,O  
utlineBrainstem_right,...  
    LC_left,LC_right,BottomBrainstem];  
  
save(['/path/where/the/folder/the/data/will/be/saved/EuclideanDistance_land  
marks.mat'],'ED_export')  
  
disp('ED calc done')
```

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

```
%% calculate Euclidean distance: 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 mask'});

%% 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_'
'_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

%% ED from the centroid point, 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
        Lcentrer_mni(slices,:)= [tmp(:,1:2) MNI_slices_L(slices)];
    else
        Lcentrer_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
        Rcentrer_mni(slices,:)= [tmp(:,1:2) MNI_slices_R(slices)];
    else
        Rcentrer_mni(slices,:)= [mean(tmp(:,1:2)) MNI_slices_R(slices)];
    end

end

% now find centres of individual masks
Lcentre_indiv=[];Rcentre_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
        Lcentre_indiv{id,1}(sl,:)= [tmp(:,1:2) slicesL(sl)];
    else
        Lcentre_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
        Rcentre_indiv{id,1}(sl,:)= [tmp(:,1:2) slicesR(sl)];
    else
        Rcentre_indiv{id,1}(sl,:)= [mean(tmp(:,1:2)) slicesR(sl)];
    end
end

%% calculate Euclidian distance of 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 = Lcentre_indiv{id,1}(:,3);
    inds1=ismember(MNI_slices_L,Lslc_tmp,'rows');
    inds2=ismember(Lslc_tmp,MNI_slices_L,'rows');

    A = Lcentrer_mni(inds1,:);
    B = Lcentre_indiv{id,1}(inds2,:);
    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 = Rcentre_indiv{id,1}(:,3);
    inds1=ismember(MNI_slices_R,Rslc_tmp,'rows');
    inds2=ismember(Rslc_tmp,MNI_slices_R,'rows');

```

```

A = Rcenter_r_mni(indslc1,:);
B = Rcentre_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

% 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

- 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.
- 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.