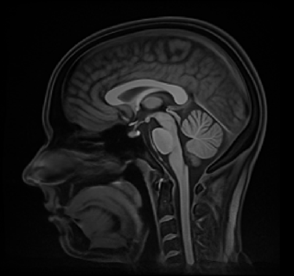
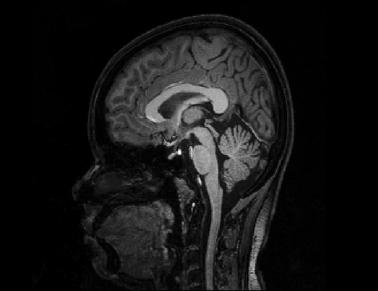
**Quality assessment of spatial transformation of locus coeruleus (LC) functional and structural magnetic resonance imaging (MRI) data**

1. Introduction
   1. Summary of spatial transformation procedure
   2. Overview of landmark assessment procedure for functional MRI data
   3. The background and the goal of this procedure
2. Protocol
   1. Preparation
      1. Required data
      2. Required software
   2. Procedure
      1. Selecting landmarks
      2. Drawing landmarks
      3. Aggregating single-subject landmarks for preliminary inspection
      4. Statistical quality assessment of aggregated landmarks of transformed functional images
      5. Statistical quality assessment of transformed structural images
3. Codes
   1. The spatial transformation pipeline (a shell script)
   2. Generate Nifti landmark overlay image (a MATLAB script)
   3. Generate video of transformed mean functional or structural images (a MATLAB script)
   4. Calculate the Euclidean distance of single-subject landmarks (a MATLAB script)
   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. **Introduction**
   1. **Summary of spatial transformation procedure**



(b) Study template



(a) MNI

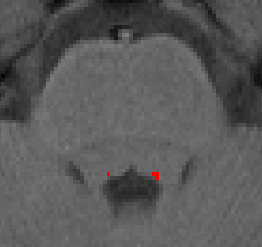
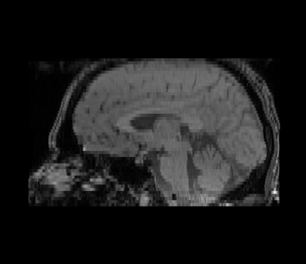
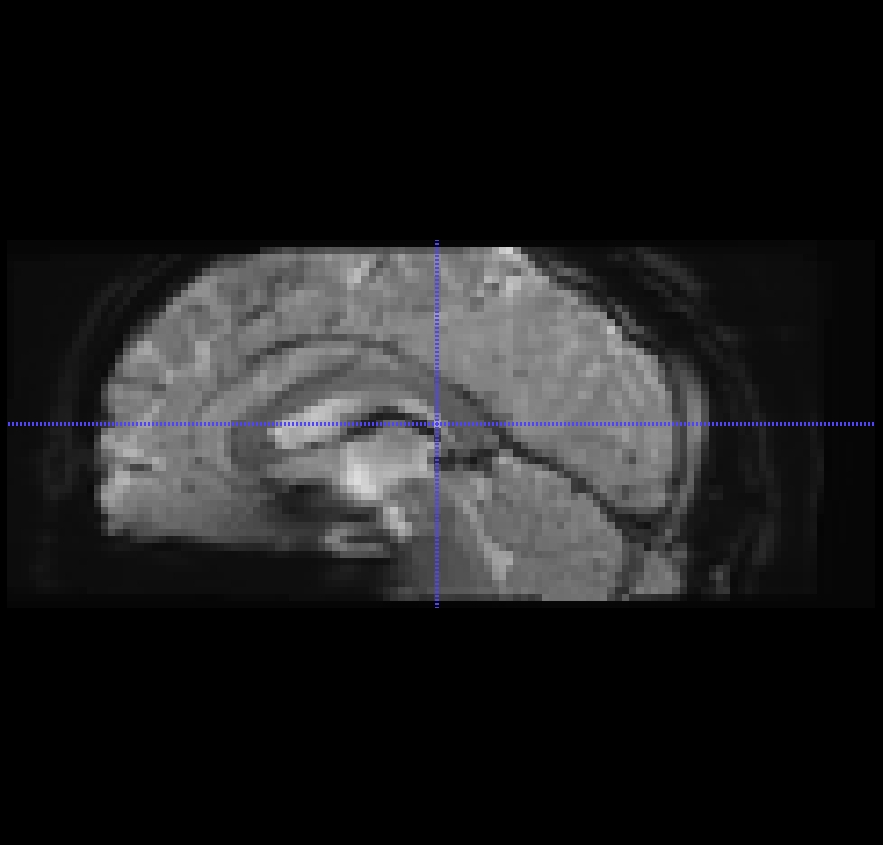
(c) Whole-brain sMRI

**(1)**

**(2)**

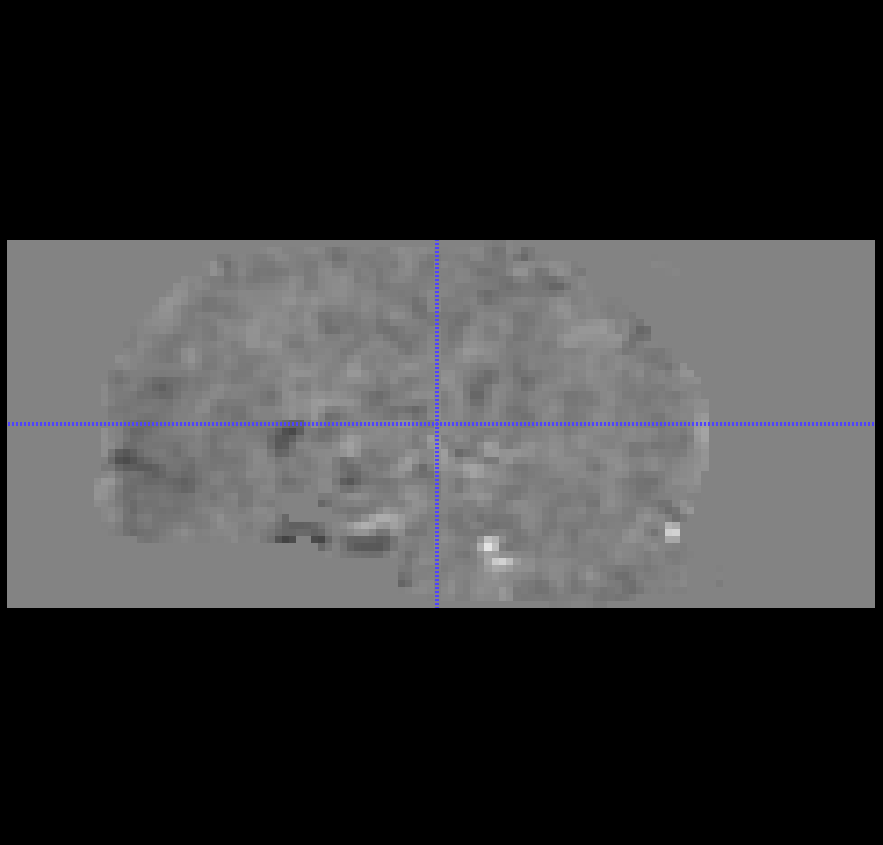
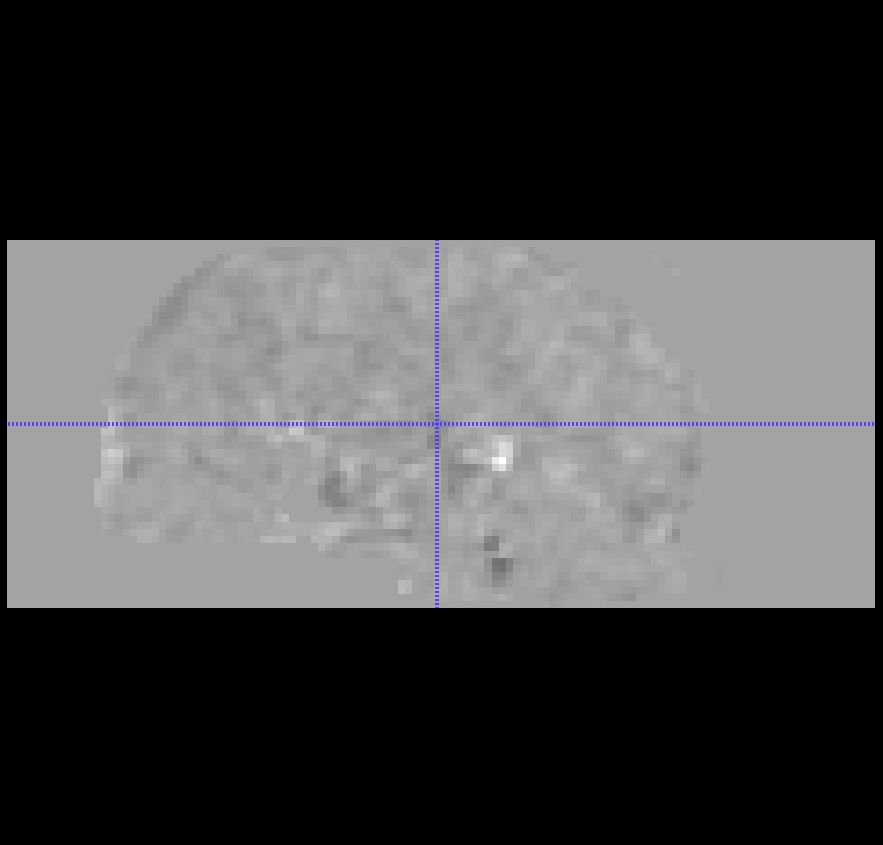
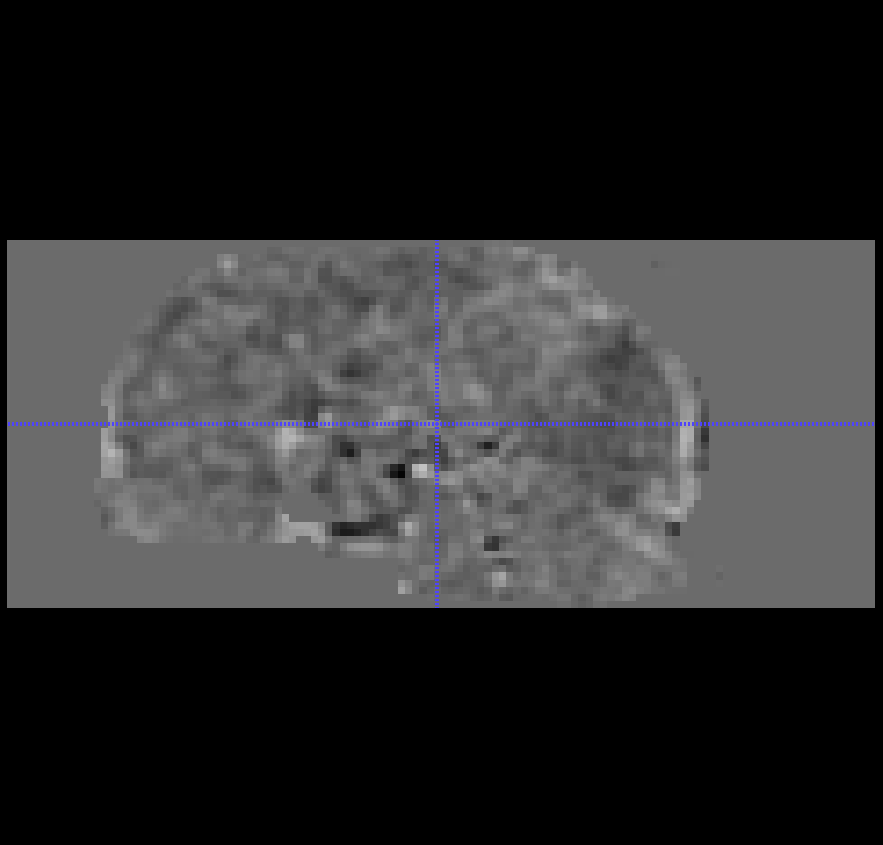
**(5-1)**

**(5-2)**



(f) LC-sensitive sMRI

(g) LC mask



(e) Statistical

maps

**(4-1)**

**(4-2)**

(d) Mean fMRI

**(3)**

Previous to the landmark-based spatial transformation quality assessment, 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). As explained in detail in the section 2.5 of the main text, single-subject functional and structural images in native space (c-g) were transformed into MNI space (a). The code for the spatial transformation procedure described above is in the section 3.b of this manual.

* 1. **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. **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.

1. **Protocol**
   1. **Preparation**
      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.

* + 1. **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

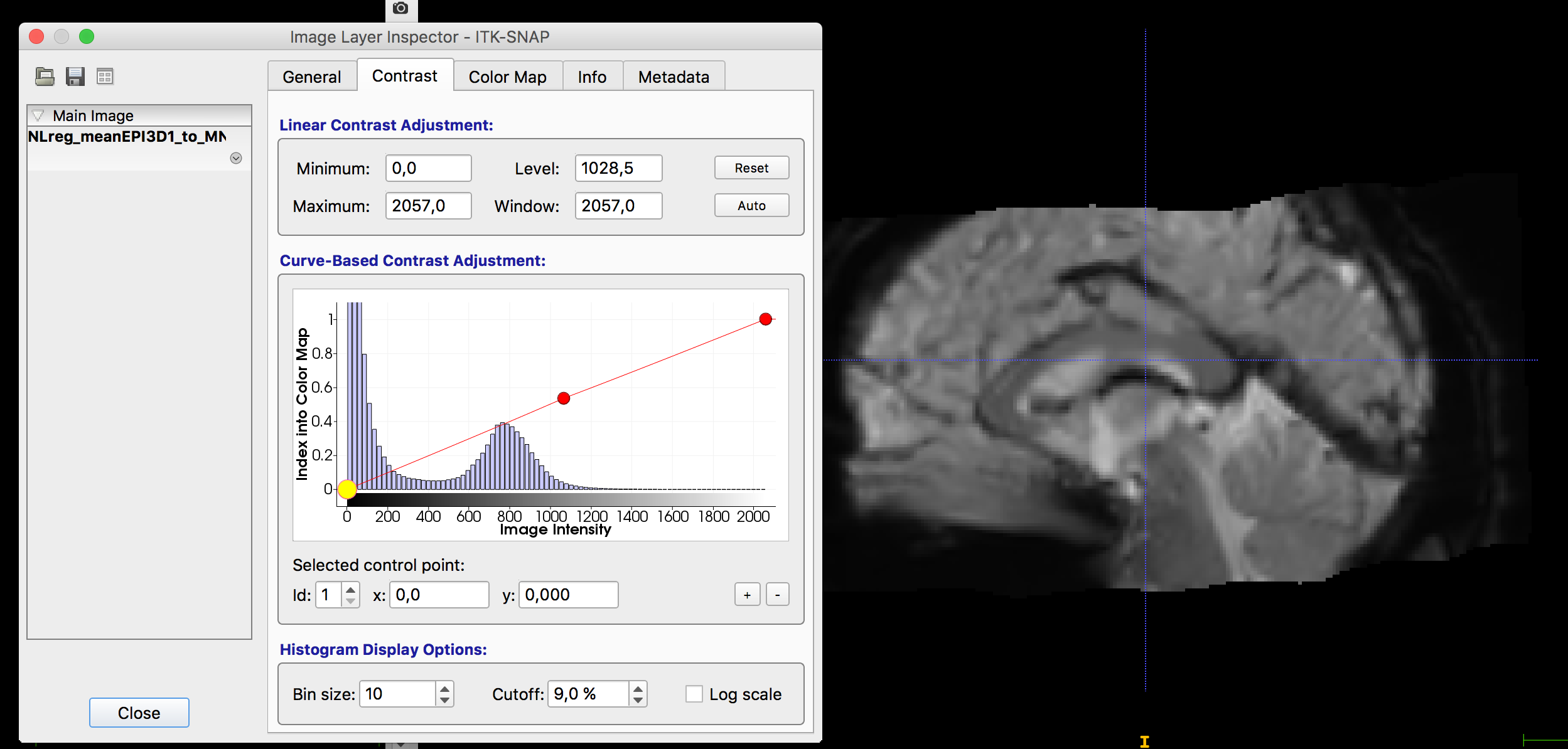
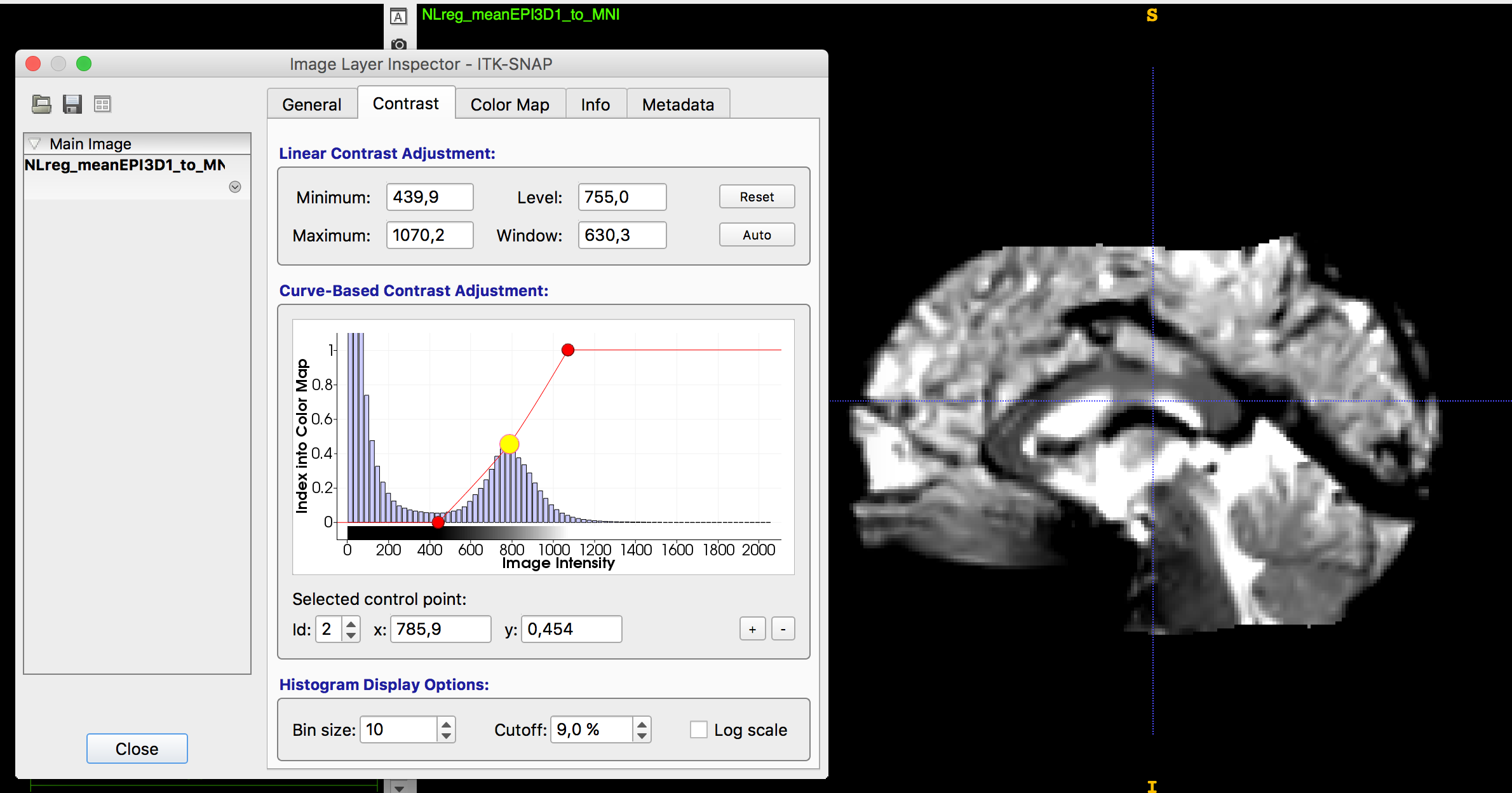
- MATLAB (version later than 2013b, Mathworks, Sherborn, MA, USA), or any computational programmes that can read Nifti images

The example MATLAB codes for generating the overlay video and statistical assessment is in section 4.

* 1. **Procedure**
     1. **Selecting landmarks**

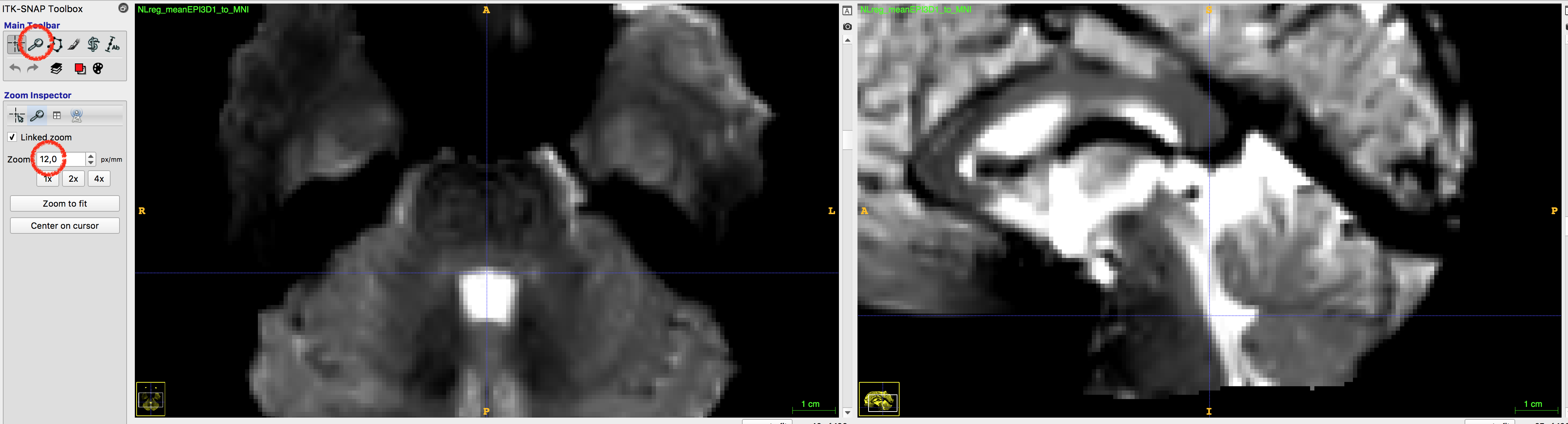
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Axial view** | **Sagittal view** | **Coronal view** |
| **Top of the brainstem and Nuclei Ruber**  **(slice 71)** | A picture containing blur, blurry  Description automatically generated | A picture containing text, blur  Description automatically generated | A picture containing blur  Description automatically generated |
| **LC and the brainstem outline**  **(slice 60)** | A picture containing blur  Description automatically generated |  | A picture containing blur  Description automatically generated |
| **Lower posterior border of brainstem**  **(slice 50)** | A picture containing blur  Description automatically generated |  | A picture containing blurry, blur  Description automatically generated |

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.



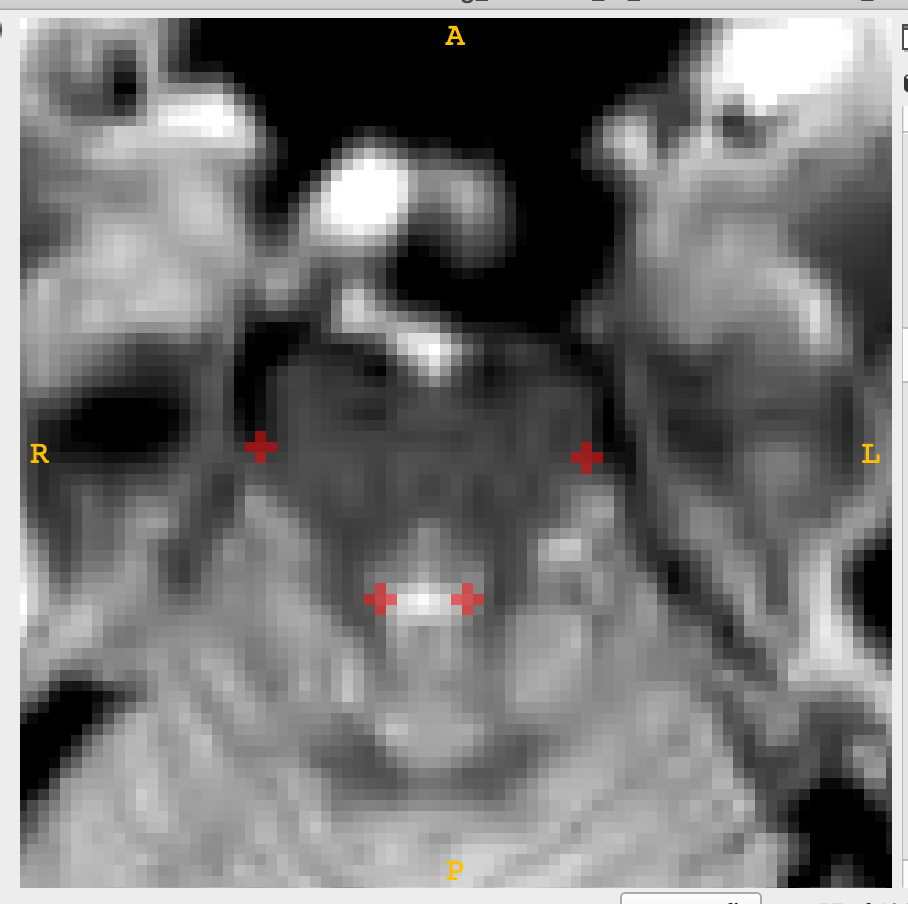
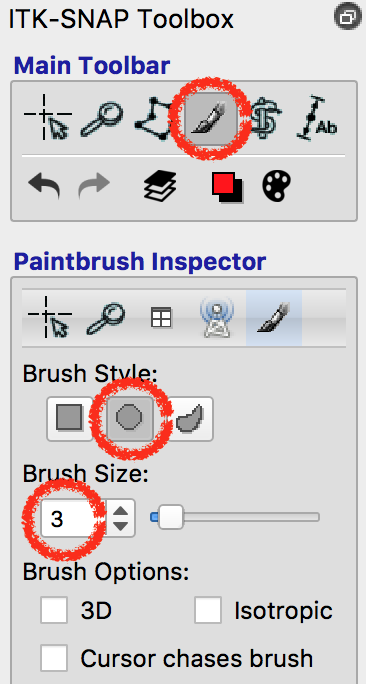
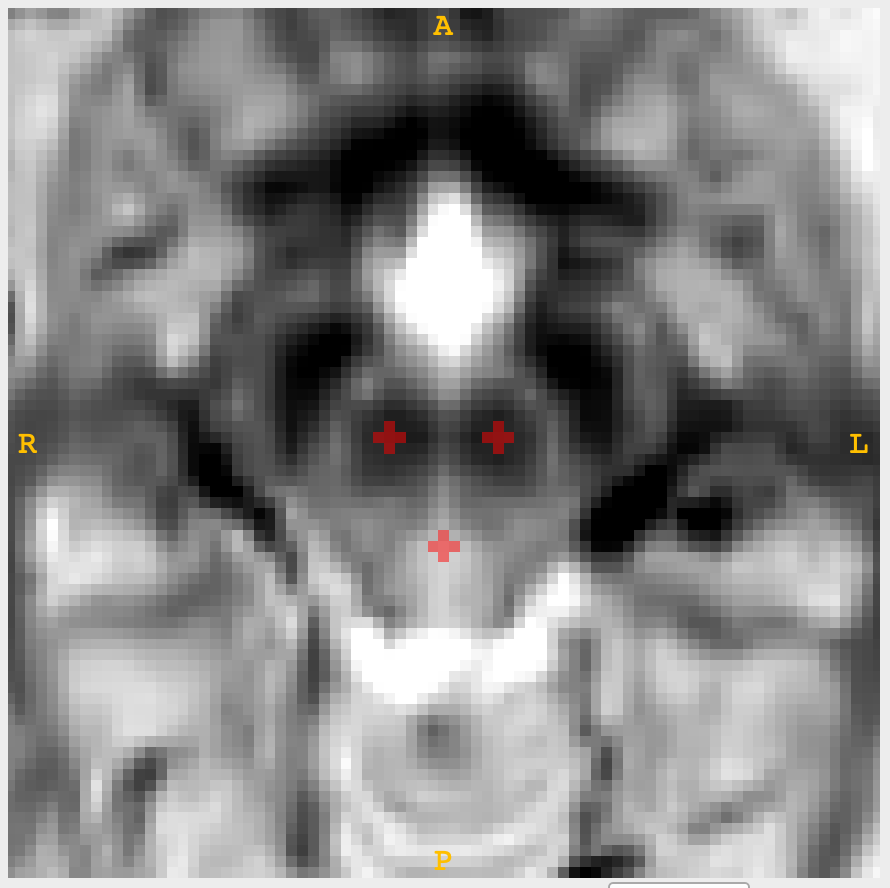
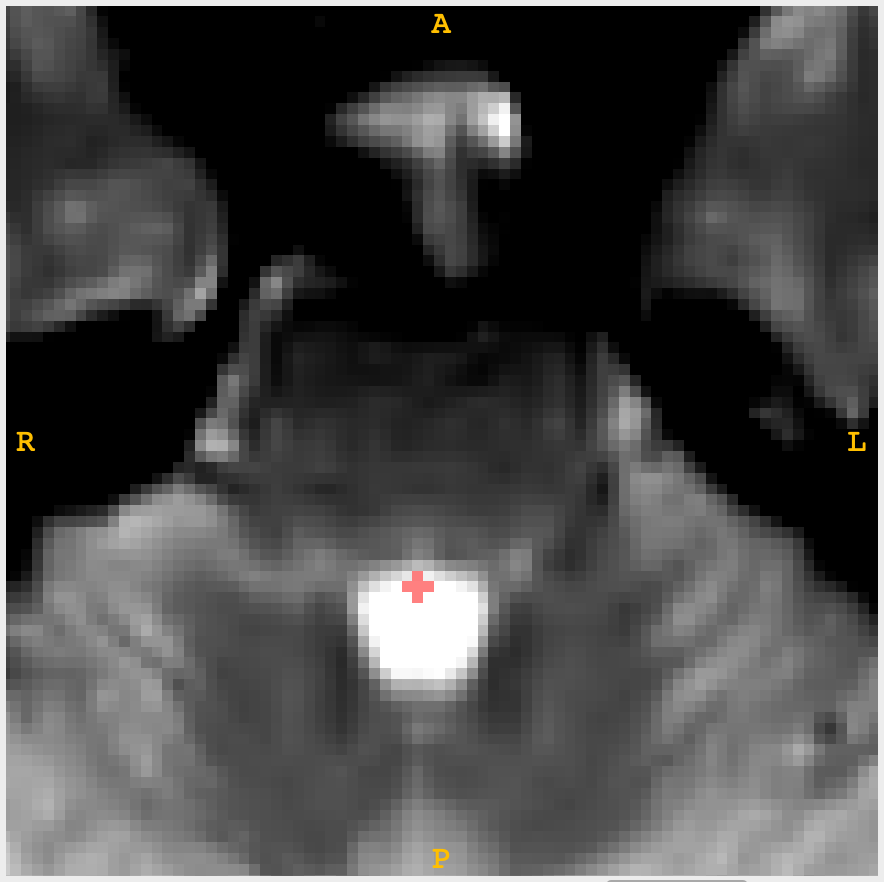
* + 1. **Drawing landmarks**

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.

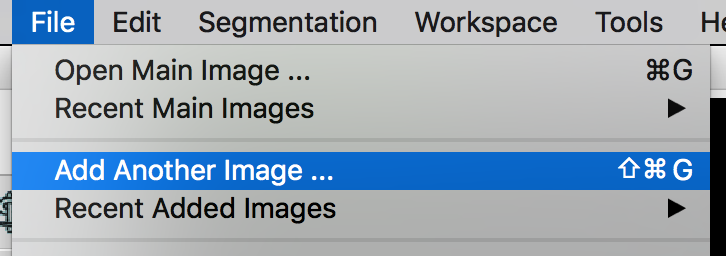
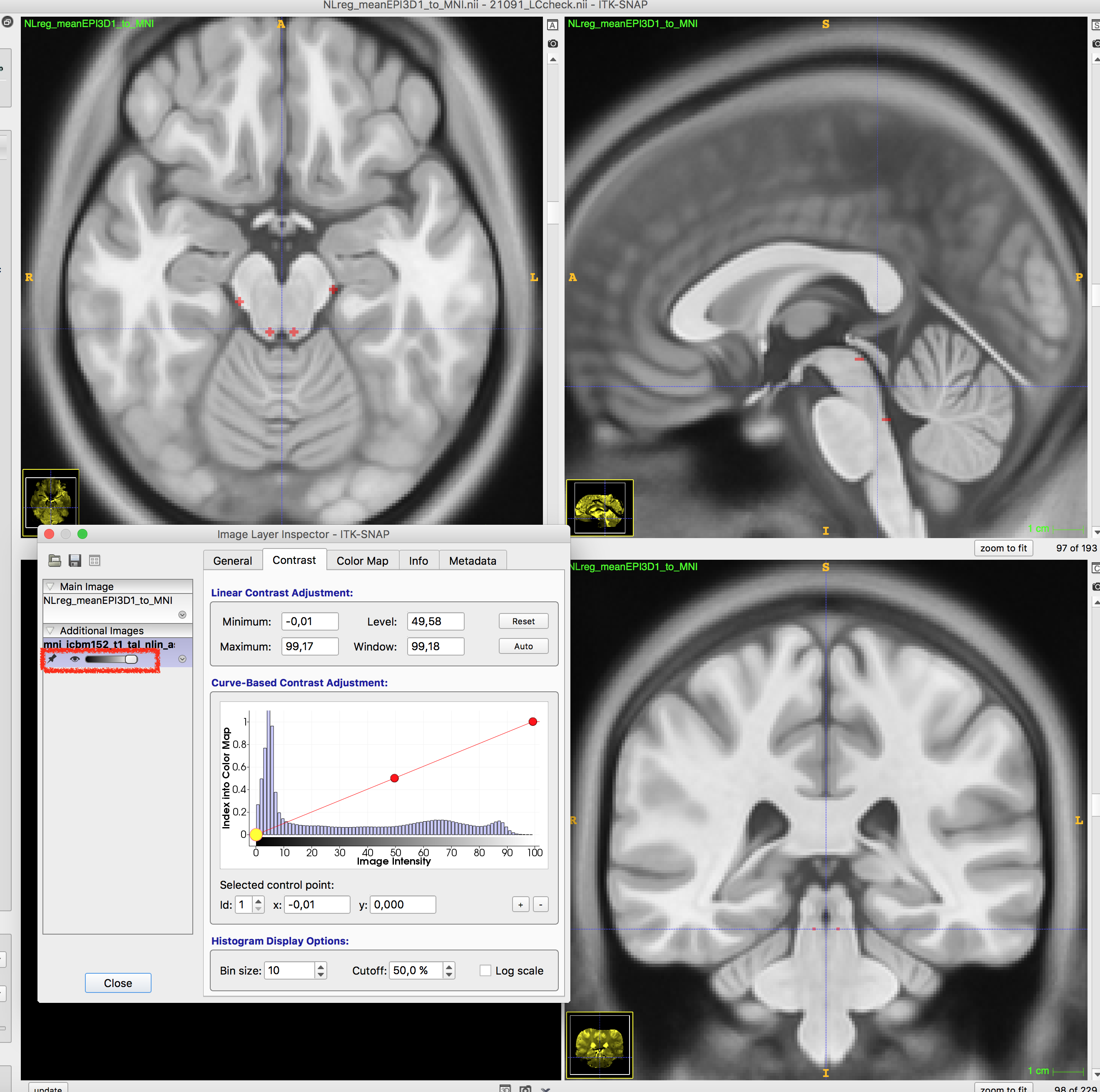
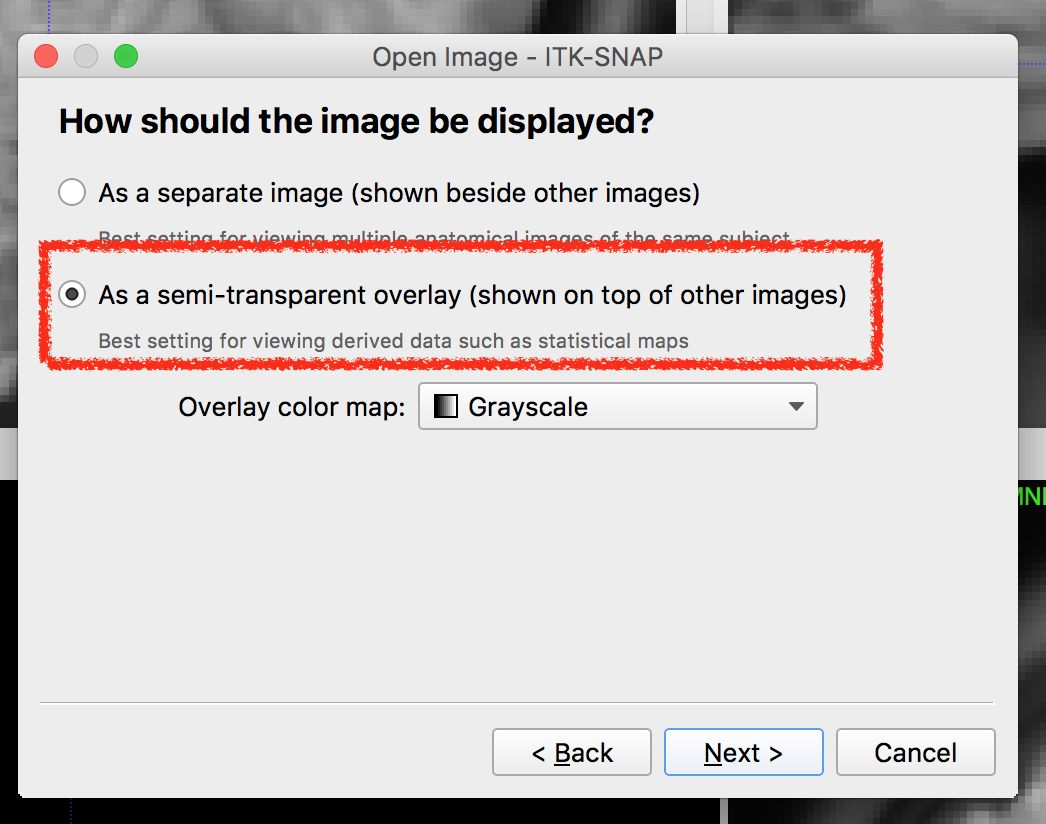


Graphical user interface, text, application, email

Description automatically generatedThen, 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.



* + 1. **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.

|  |  |  |
| --- | --- | --- |
| A picture containing white  Description automatically generated | A picture containing text, light, device, traffic light  Description automatically generated | A picture containing indoor, blurry  Description automatically generated |

* + 1. **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.

* + 1. **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.

1. **Codes**
   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

# ------------------------------------------------------ #

* 1. **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);

* 1. **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 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;

* 1. **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.OutlineBrainstem\_left-MNI.OutlineBrainstem.left).^2).^0.5;

EuclideanDistances\_indv{subj,1}.OutlineBrainstem.right=sum((transformed\_landmark\_coords{subj,1}.MidSlice.OutlineBrainstem\_right-MNI.OutlineBrainstem.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,OutlineBrainstem\_right,...

LC\_left,LC\_right,BottomBrainstem];

save(['/path/where/the/folder/the/data/will/be/saved/EuclideanDistance\_landmarks.mat'],'ED\_export')

disp('ED calc done')

* 1. **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\_icbm152\_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\_' 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

%% 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

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);

indslc1=ismember(MNI\_slices\_L,Lslc\_tmp,'rows');

indslc2=ismember(Lslc\_tmp,MNI\_slices\_L,'rows');

A = Lcentrer\_mni(indslc1,:);

B = Lcentre\_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 = Rcentre\_indiv{id,1}(:,3);

indslc1=ismember(MNI\_slices\_R,Rslc\_tmp,'rows');

indslc2=ismember(Rslc\_tmp, MNI\_slices\_R,'rows');

A = Rcentrer\_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

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))