Complex Networks fMRI Preprocessing Pipeline

Mika Rubinov
Prantik Kundu

Brain Mapping Unit

The preprocessing is in six parts

The preprocessing is in six parts

Signal preprocessing

1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. Removal of noise signal

Network construction

5. Construction of nodes
6. Construction of links

We use the
AFNI, FSL and WMTSA
software packages

We do most analyses with AFNI and FSL

- Two C software packages for processing MRI data.

- AFNI: Analysis of Functional NeuroImages. Made by the NIH.

- FSL: FMRIB Software Library. Made by the FMRIB in Oxford.

We do most analyses with AFNI and FSL

1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. Removal of noise signal

5. Construction of nodes
6. Construction of links

We construct links with WMTSA

- WMTSA: Wavelet Methods for Time-Series Analysis

- A Matlab and R program for computing frequency-band specific “wavelet” correlations.

We construct links with WMTSA

1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. Removal of noise signal

5. Construction of nodes
6. **Construction of links**

An overview of the main steps

Preprocessing steps

1. **Preprocessing of anatomical images**
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. Removal of noise signal

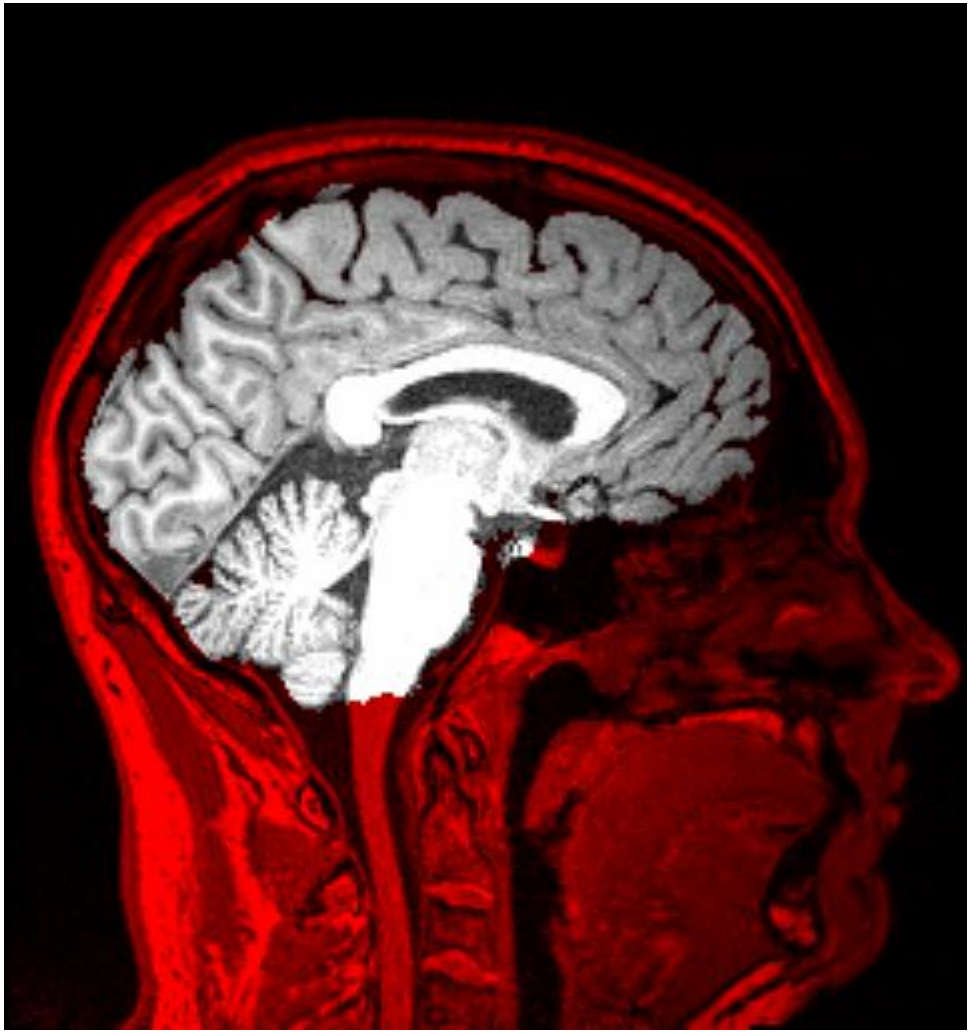
5. Construction of nodes
6. Construction of links

Skull-stripping in anatomical image

```
## Get skull-strip mask of anatomical image
3dSkullStrip \
    -o_ply anatomical_skullstrip_mask.nii.gz \
    -input anatomical_image.nii.gz \

## Apply skull-strip mask to anatomical image
3dcalc \
    -prefix anatomical_skullstrip.nii.gz \
        -expr 'a*step(b)' \
        -b anatomical_skullstrip_mask.nii.gz \
    -a anatomical_image.nii.gz \
```

Skull-stripping in anatomical image



Preprocessing steps

1. Preprocessing of anatomical images
2. **Preprocessing of functional images**
3. Anatomical standardization of functional images
4. Removal of noise signal

5. Construction of nodes
6. Construction of links

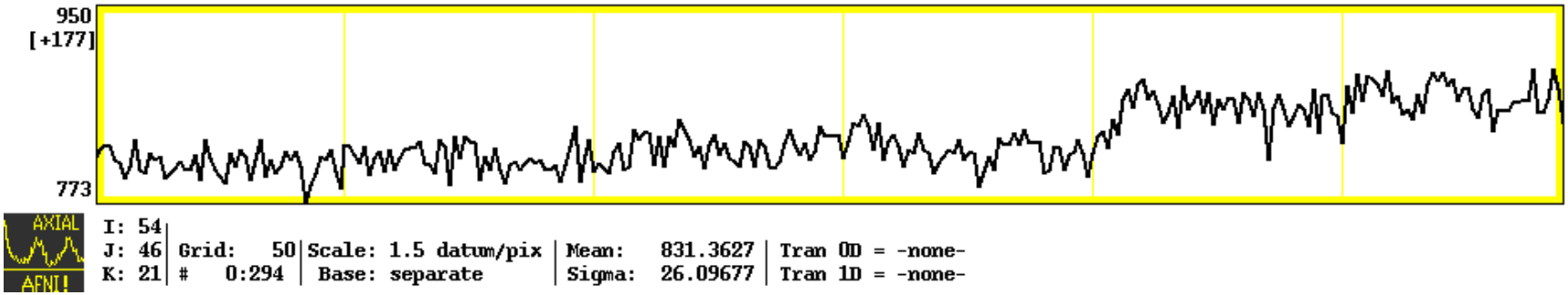
Slice-timing correction of functional image

```
## Correct functional image for slice-timing
3dTshift \
    -prefix ${func}_st_F.nii.gz \
        -tpattern altplus \
    ${func}_ro_F.nii.gz \
```

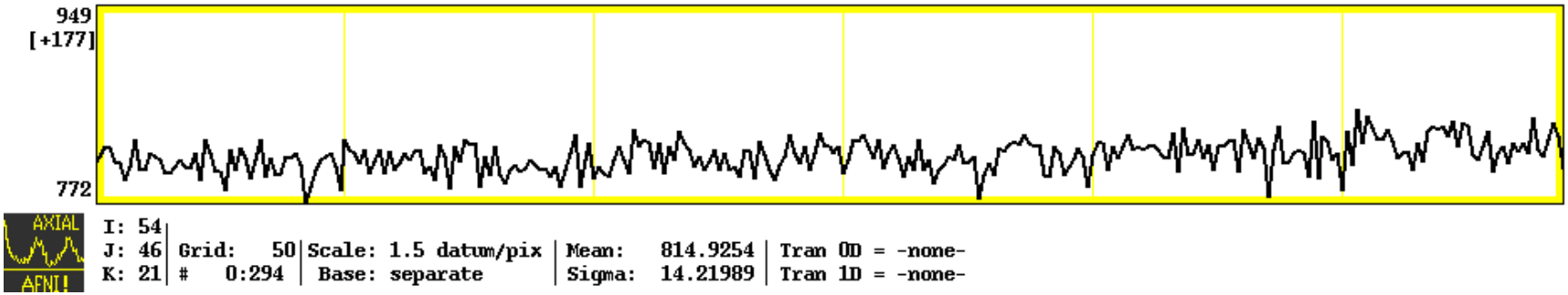
Motion correction (realignment) of functional image

```
## Correct functional image for motion
3dvolreg \
    -prefix ${func}_mc_F.nii.gz \
        -1Dfile ${func}_motion.1D \
        -Fourier -twopass -zpad 4 \
        -base ${func}_st_F.nii.gz[$ind] \
    ${func}_st_F.nii.gz \
```

Motion correction (realignment) of functional image



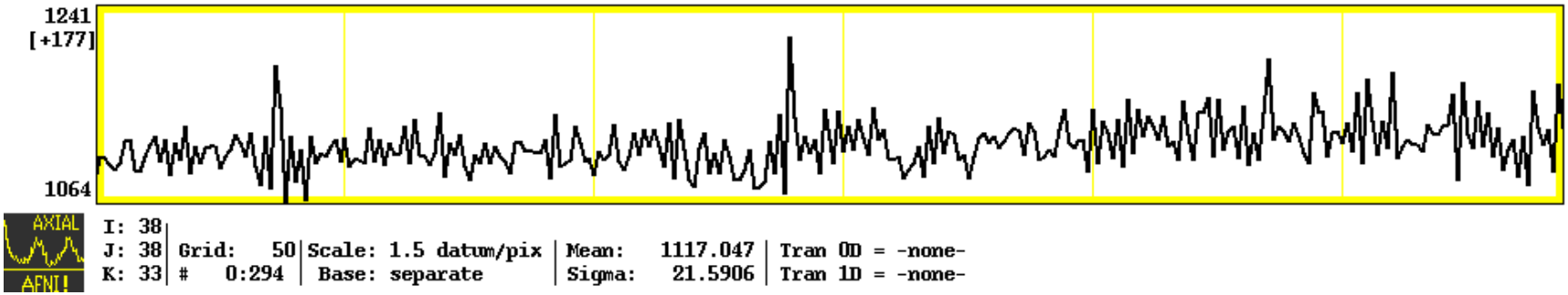
Motion correction (realignment) of functional image



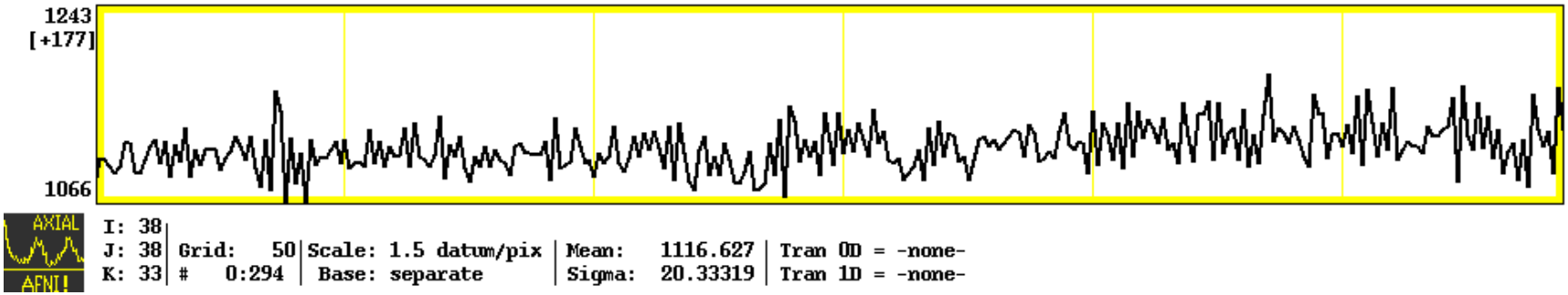
Despiking of functional image

```
## Despike functional image
3dDespike \
    -prefix ${func}_ds_F.nii.gz \
        -ssave ${func}_spikiness.nii.gz \
    ${func}_mc_F.nii.gz \
```

Despiking of functional image



Despiking of functional image



Preprocessing steps

1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. **Anatomical standardization of functional images**
4. Removal of noise signal

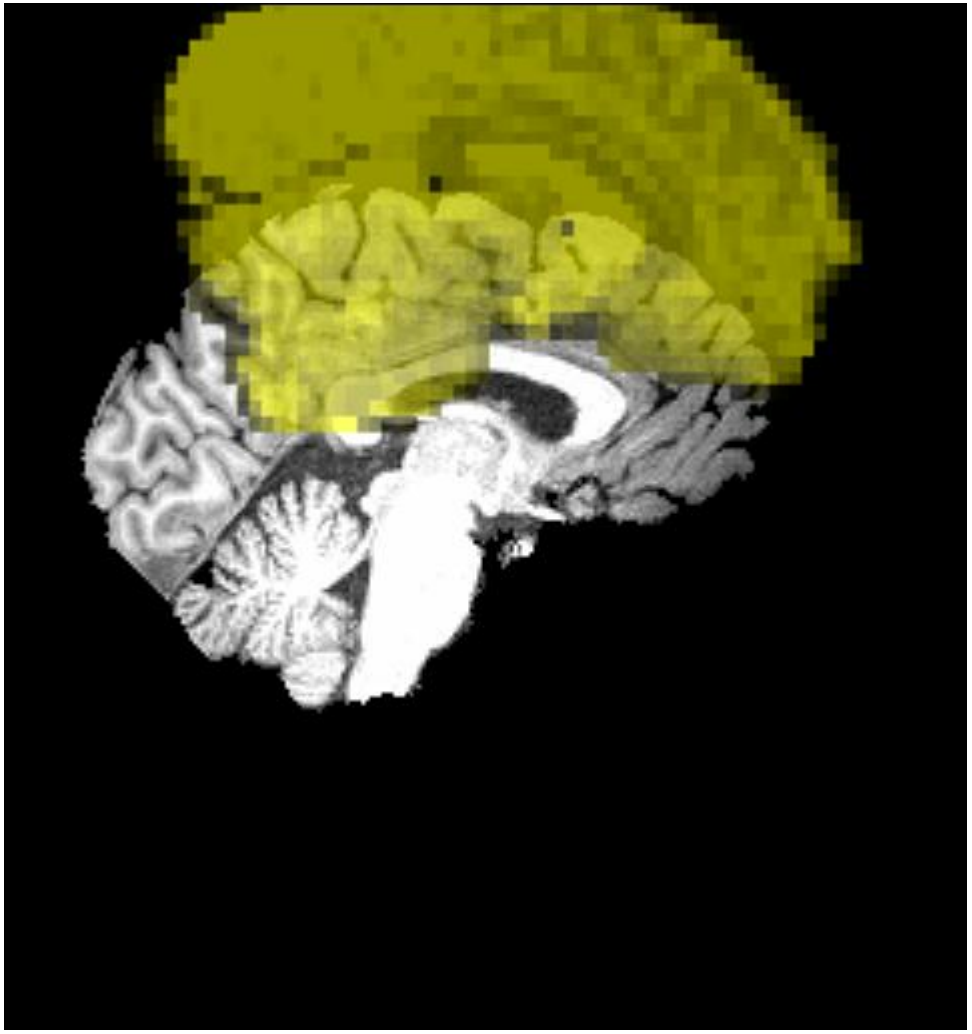
5. Construction of nodes
6. Construction of links

Registration of functional image to high-resolution anatomical image

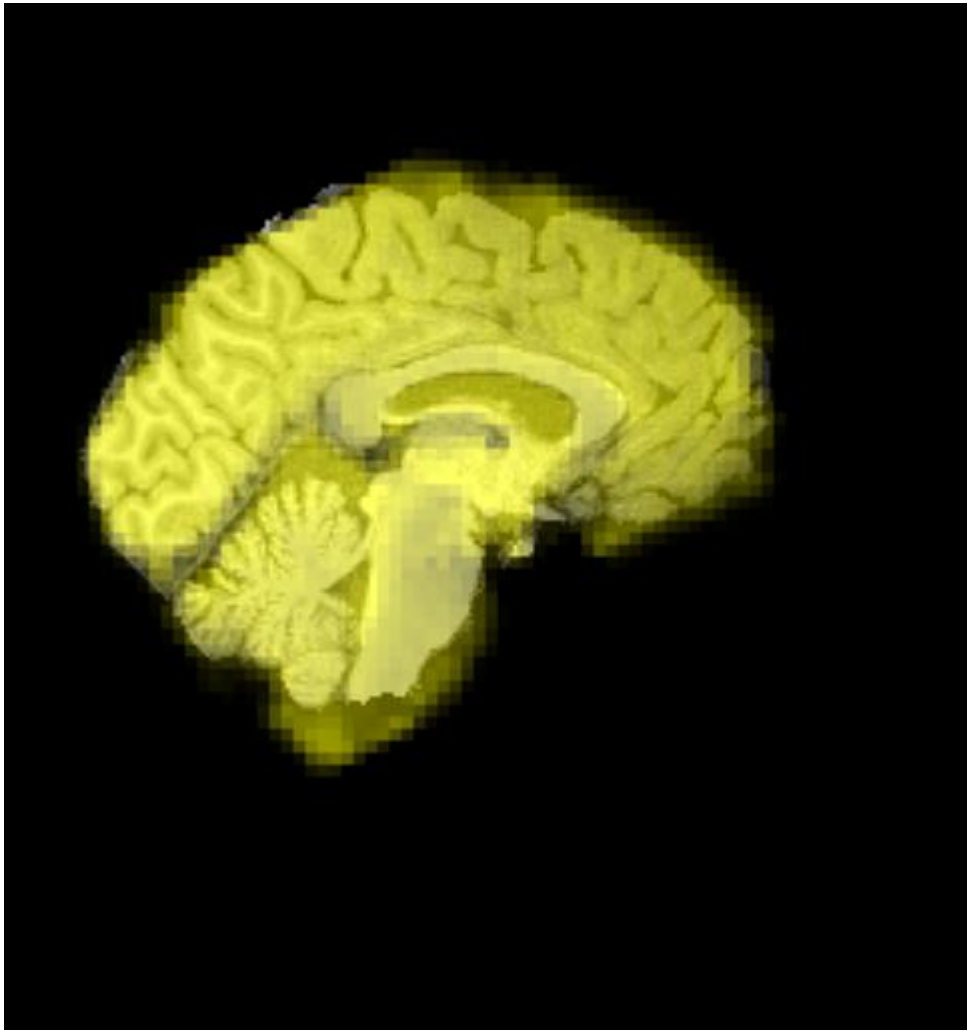
```
## Get functional-to-anatomical image registration
flirt \
    -omat func2anat.mat \
        -cost corratio -dof 12 -interp trilinear \
        -ref ${anat}_pp_A.nii.gz \
    -in ${func}_pp_${ind}_F.nii.gz \

## Apply functional-to-anatomical image registration.
flirt \
    -out ${func}_pp_A.nii.gz \
        -interp trilinear \
        -applyxfm -init func2anat.mat \
        -ref ${anat}_pp_rs_A.nii.gz \
    -in ${func}_pp_F.nii.gz \
```

Registration of functional image to high-resolution anatomical image



Registration of functional image to high-resolution anatomical image

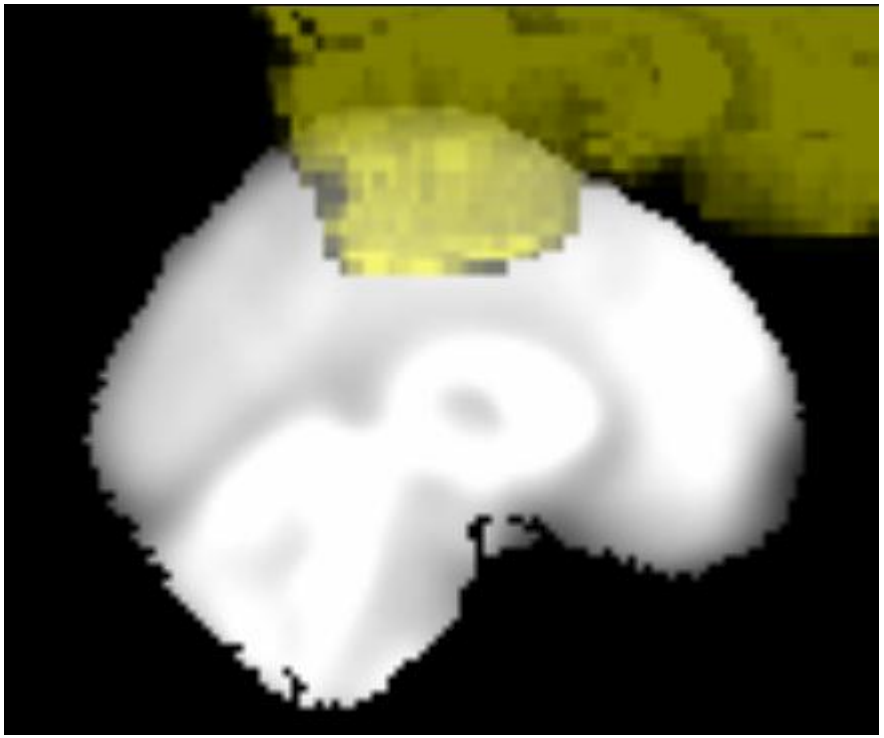


Standardization of functional image

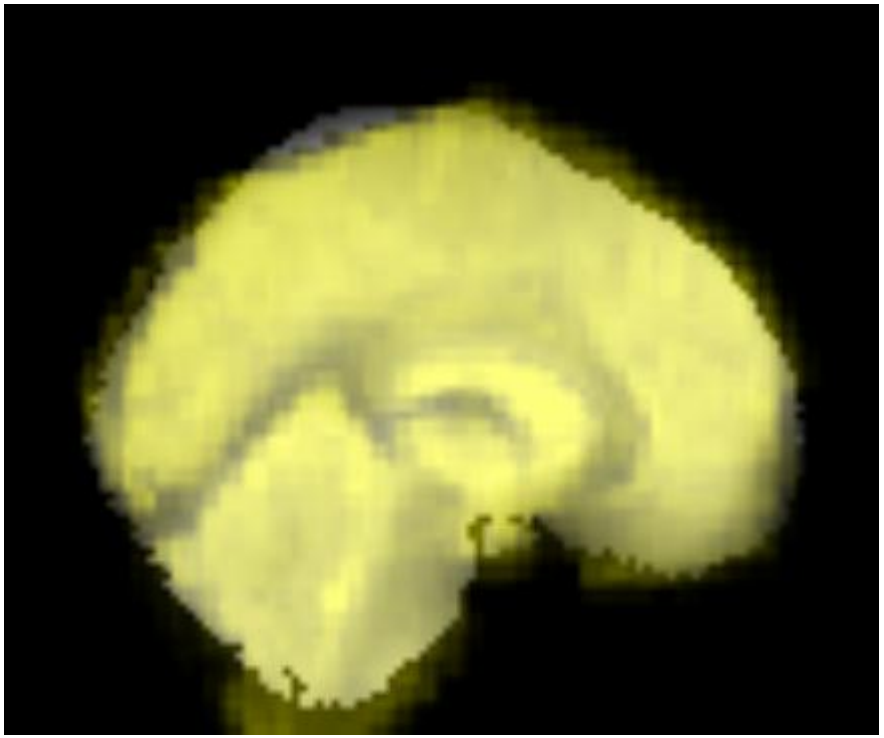
```
## Get anatomical-to-standard image registration
flirt \
    -omat anat2stnd.mat \
        -cost corratio -dof 12 -interp trilinear \
        -ref stnd_pp_T.nii.gz \
    -in ${anat}_pp_A.nii.gz \

## Get functional-to-standard image transformation
convert_xfm \
    -omat func2stnd.mat \
    -concat anat2stnd.mat func2anat.mat \
```

Standardization of functional image



Standardization of functional image



Preprocessing steps

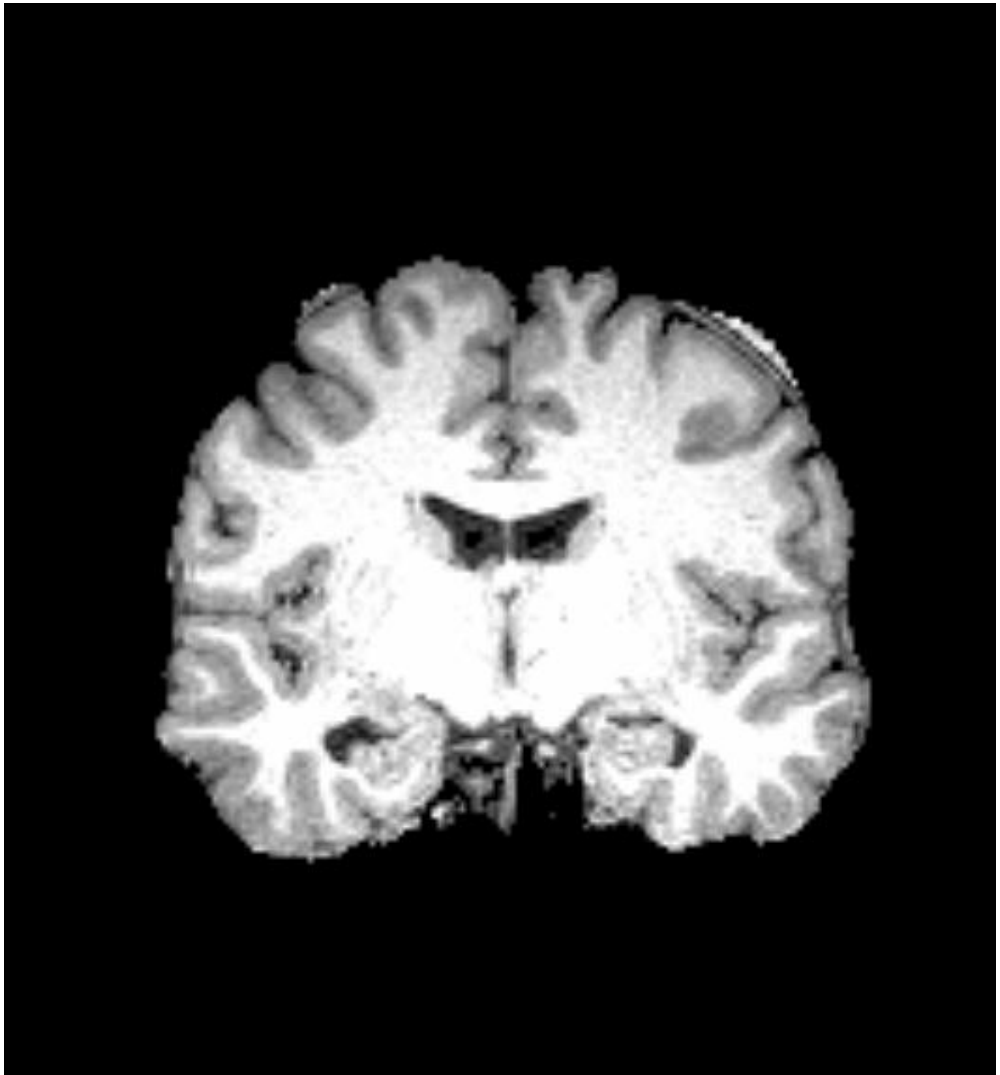
1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. **Removal of noise signal**

5. Construction of nodes
6. Construction of links

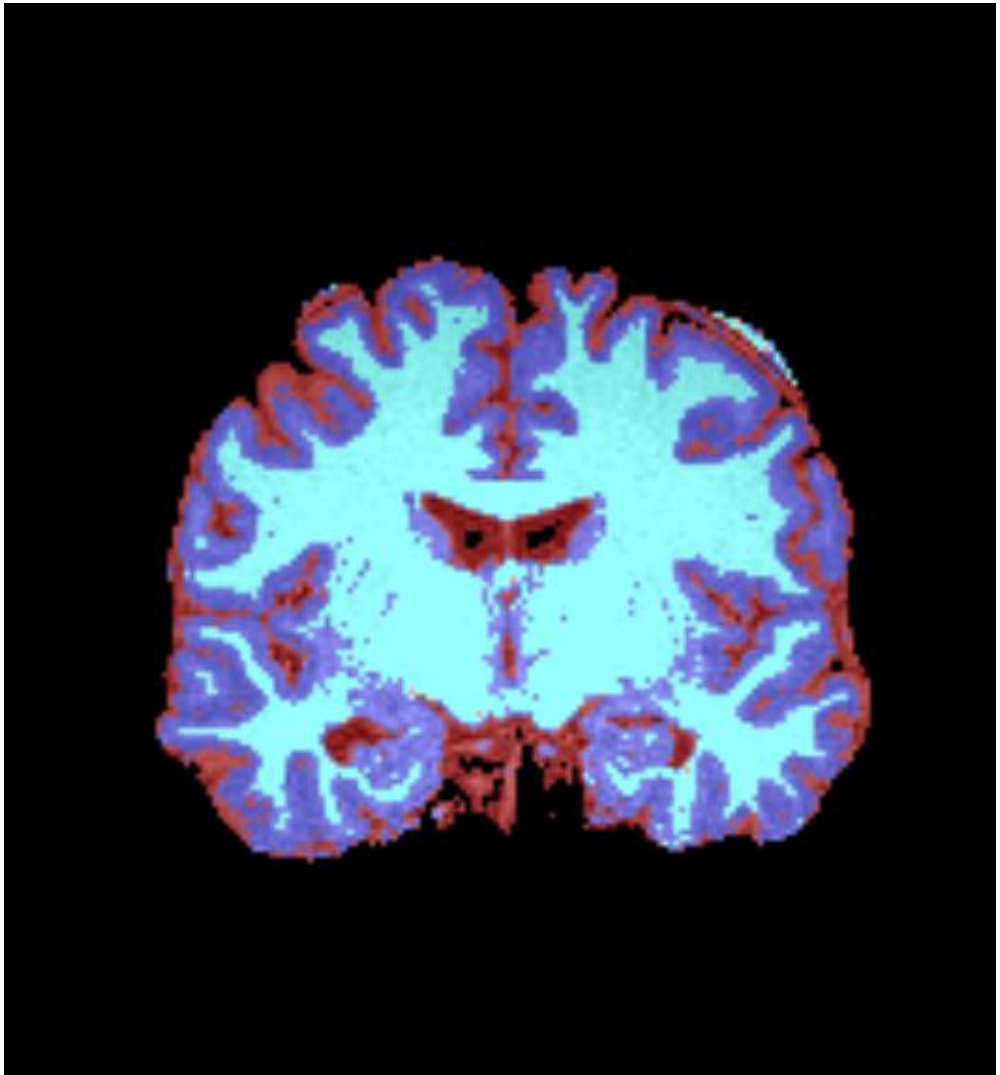
Segmentation of anatomical image into grey matter, white matter and CSF

```
## Segment anatomical image
fast \
    -o ${anat}_segm_A \
        -t 1 -n 3 -g -p \
    ${anat}_pp_A.nii.gz \
```

Segmentation of anatomical image into grey matter, white matter and CSF



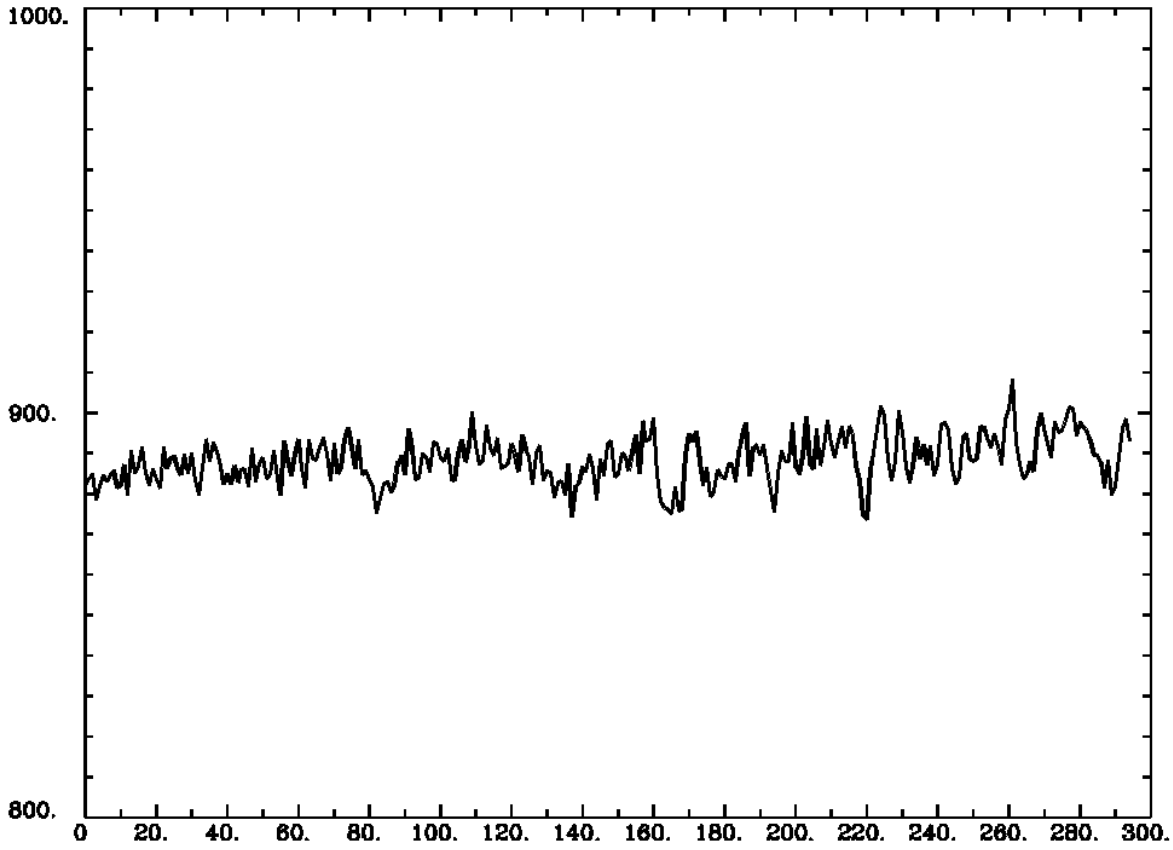
Segmentation of anatomical image into grey matter, white matter and CSF



Mean CSF signal

```
## Get mean signal of CSF segment
3dmaskave \
-quiet \
-mask ${func}_csf_A.nii.gz \
${func}_pp_A.nii.gz > ${func}_csf.1D \
```

Mean CSF Signal



Preprocessing steps

1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. **Removal of noise signal**

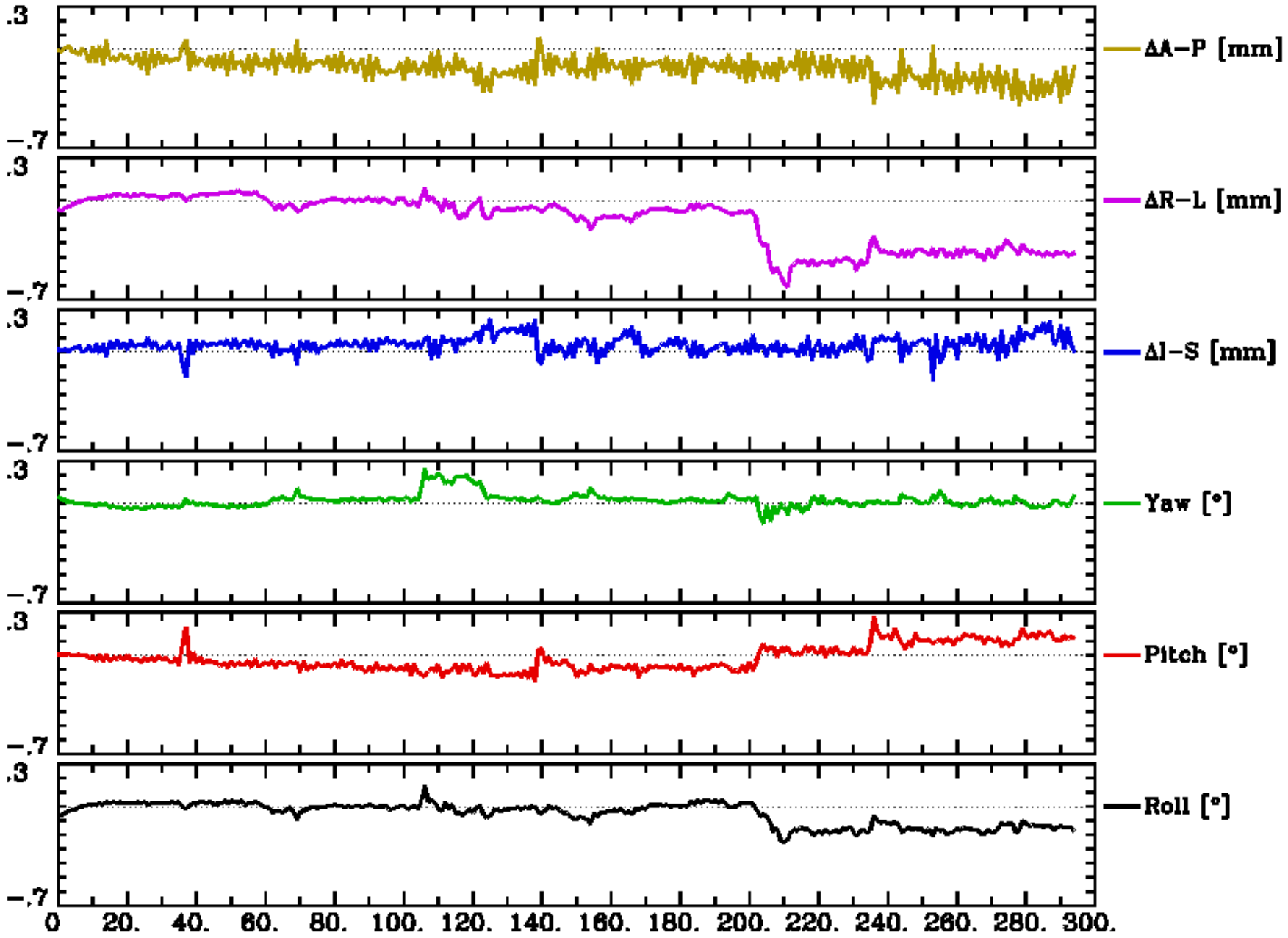
5. Construction of nodes
6. Construction of links

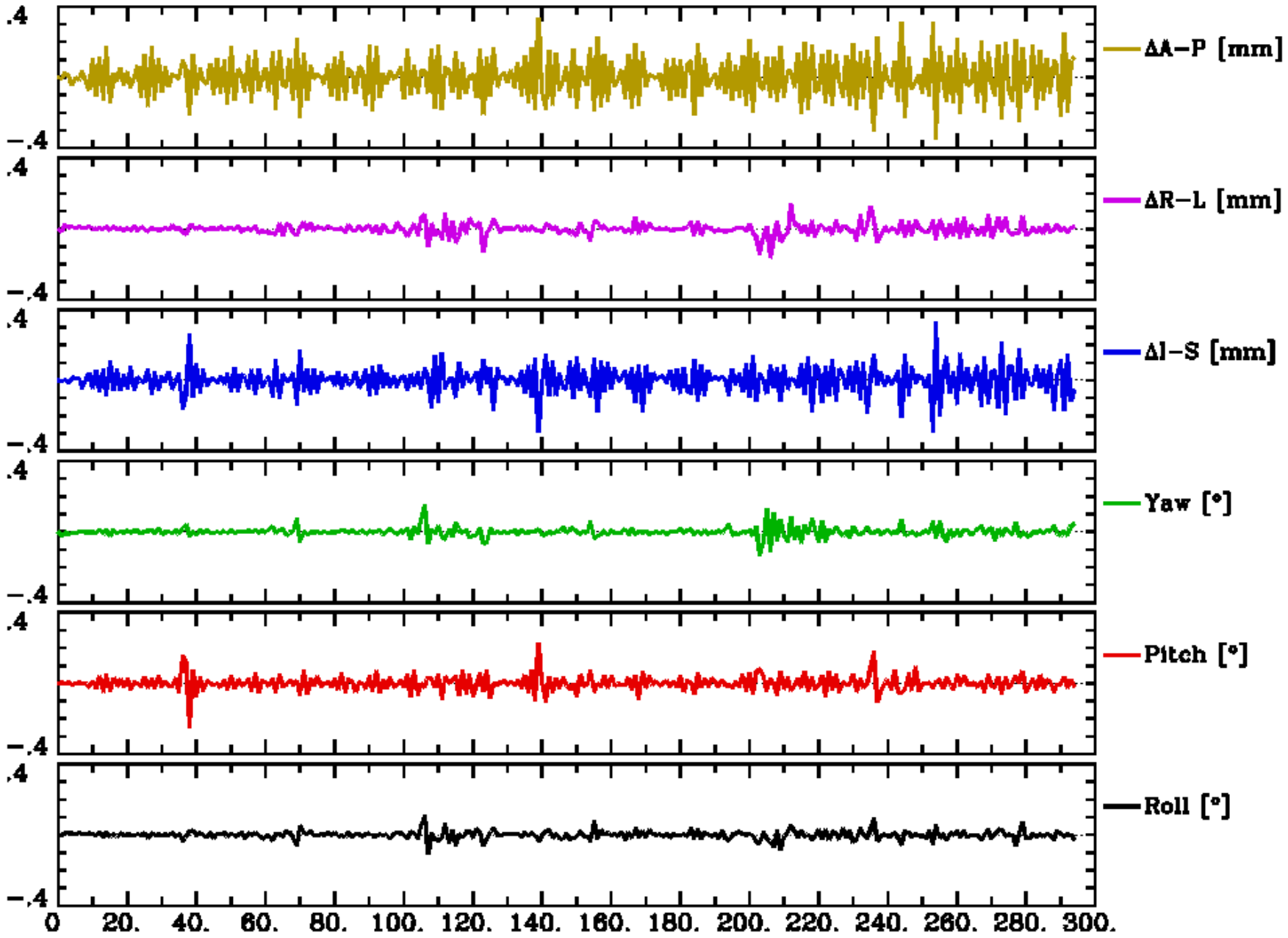
Motion parameters

```
## Correct functional image for motion
3dvolreg \
    -prefix ${func}_mc_F.nii.gz \
        -1Dfile ${func}_motion.1D \
        -Fourier -twopass -zpad 4 \
        -base ${func}_st_F.nii.gz[$ind] \
    ${func}_st_F.nii.gz \
```

Motion derivative

```
## Get motion derivative
1d_tool.py \
    -write ${func}_motion_deriv.1D \
        -derivative \
    -infile ${func}_motion.1D \
```


.3
-.7
ΔA−P [mm]
ΔR−L [mm]
Δl−S [mm]
Yaw [°]
Pitch [°]
Roll [°]
0
20.
40.
60.
80.
100.
120.
140.
160.
180.
200.
220.
240.
260.
280.
300.

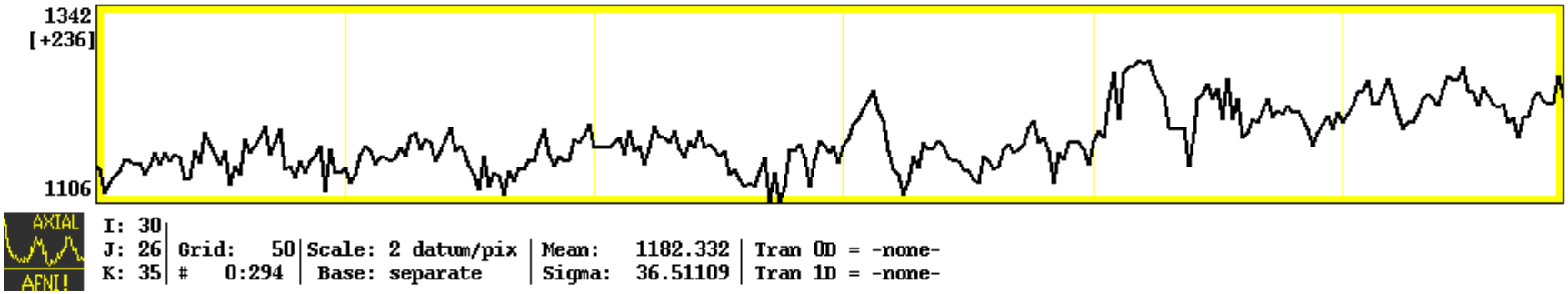

.4
−.4
ΔA−P [mm]
ΔR−L [mm]
ΔI−S [mm]
Yaw [°]
Pitch [°]
Roll [°]
0
20.
40.
60.
80.
100.
120.
140.
160.
180.
200.
220.
240.
260.
280.
300.

Removal of the noise signal

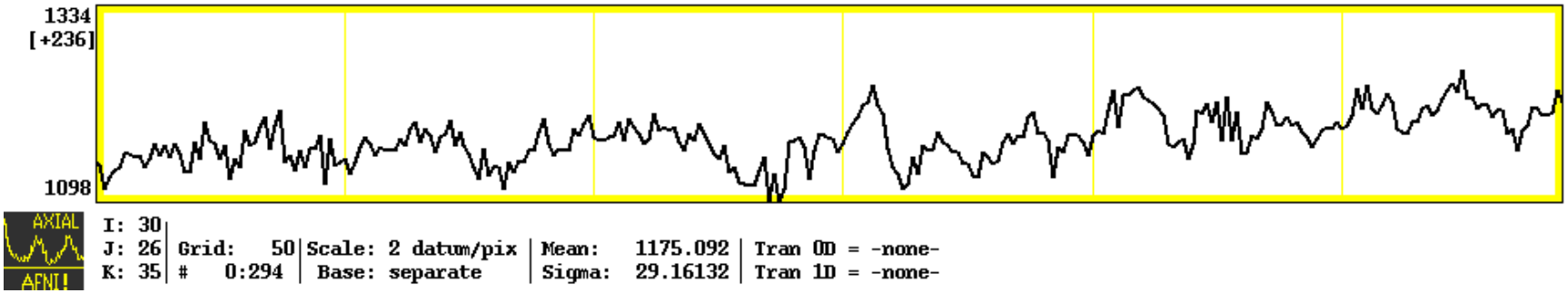
```
## Concatenate CSF signal, motion parameters, motion
derivative into 'noise signal'
1dcat
    ${func}_csf.1D ${func}_motion.1D
    ${func}_motion_deriv.1D > ${func}_noise.1D

## Regress out the 'noise signal' from functional image
3dBandpass \
    -prefix ${func}_cl_F.nii.gz \
        -mask ${func}_pp_F.nii.gz[$ind] \
        -ort ${func}_noise.1D \
        0.02 99999 \
    ${func}_pp_F.nii.gz \
```

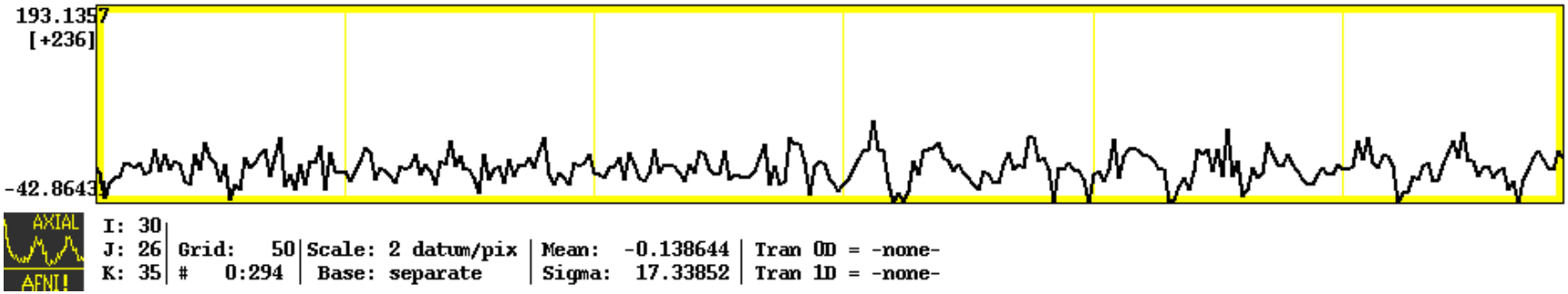
Example uncorrected signal



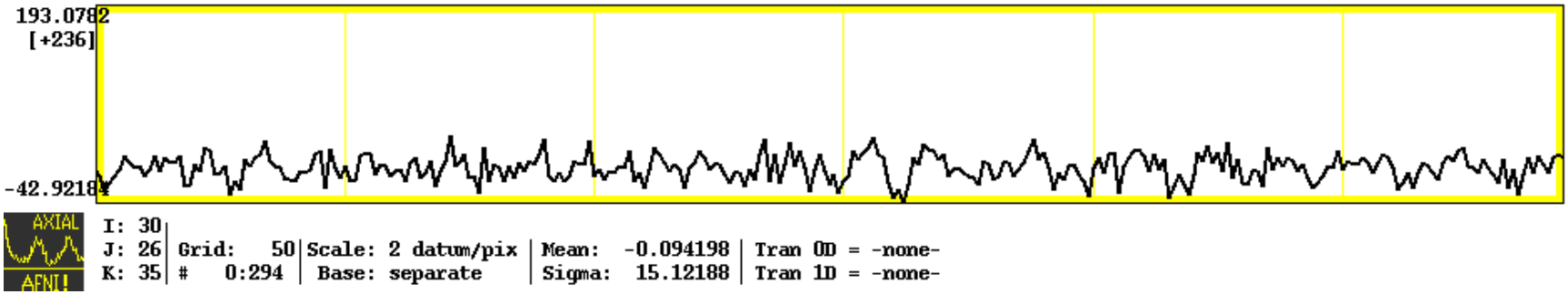
Motion-corrected and despiked signal



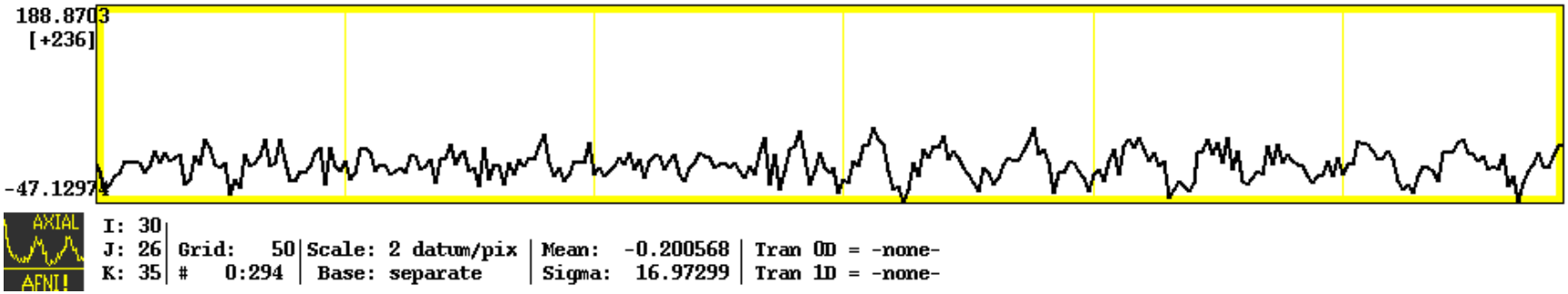
CSF signal is regressed out



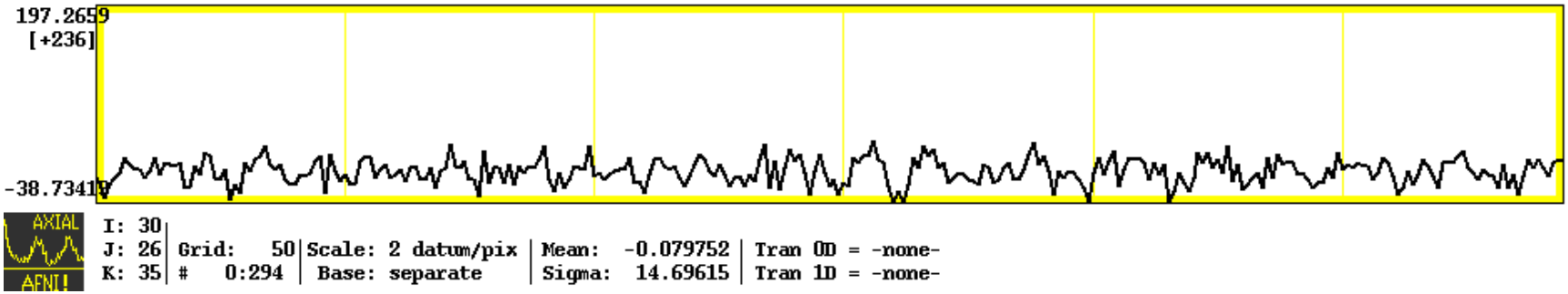
Motion parameters are regressed out



Motion derivative is regressed out



CSF signal, motion parameters and motion derivative are regressed out



Preprocessing steps

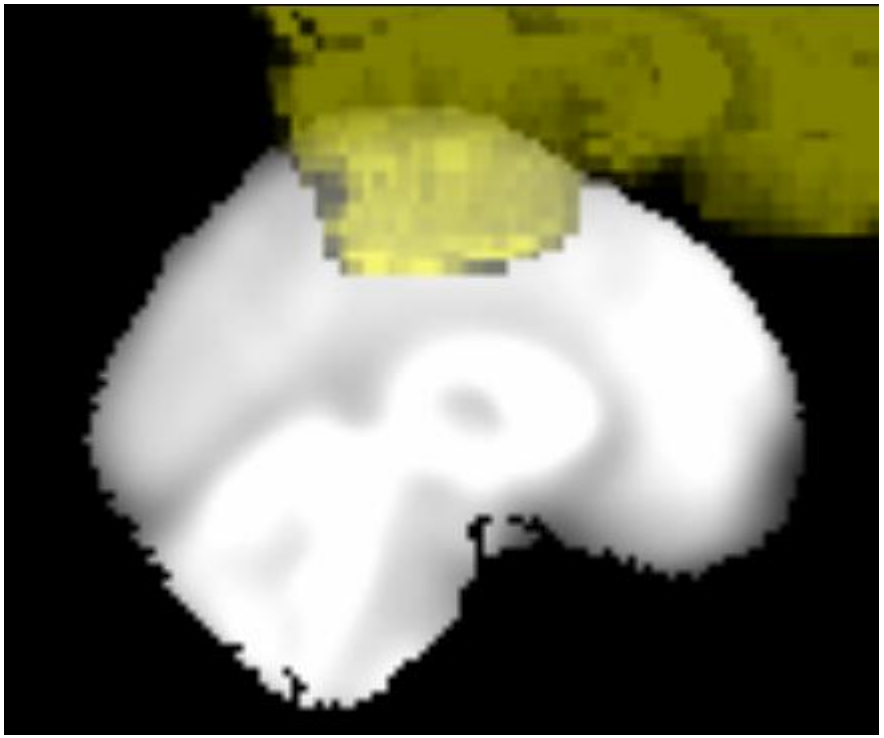
1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. Removal of noise signal

5. **Construction of nodes**
6. Construction of links

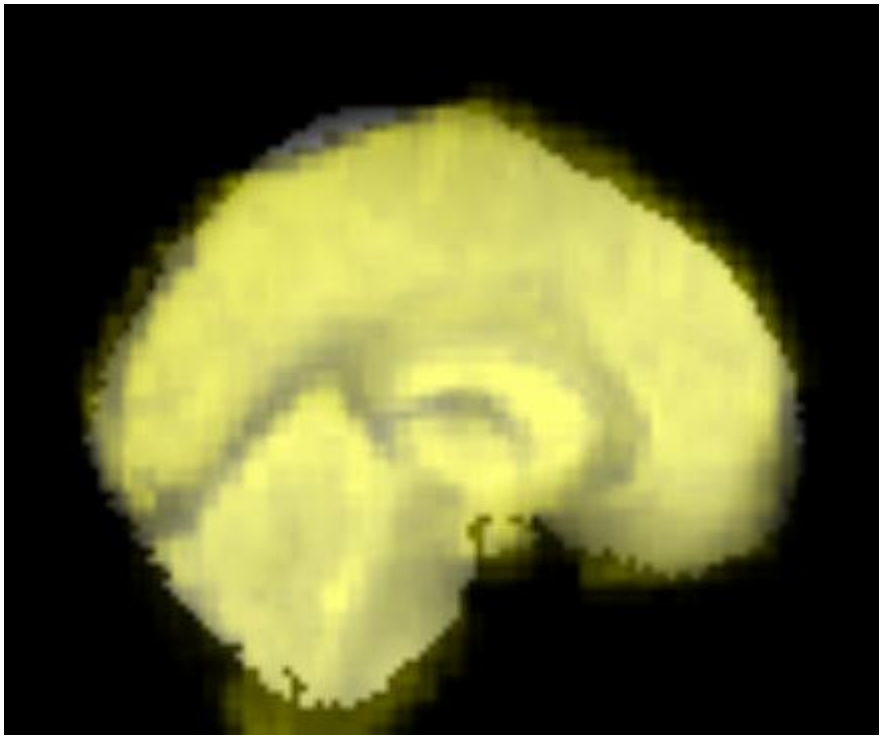
Standardization of functional image

```
## Apply functional-to-standard image
registration.
flirt \
    -out ${func}_cl_norm_T.nii.gz \
        -interp trilinear \
        -applyxfm -init func2stnd.mat \
        -ref stnd_pp_rs_T.nii.gz \
    -in ${func}_cl_norm_F.nii.gz \
```

Standardization of functional image



Standardization of functional image



Parcellation of functional image

Computation of mean signal in parcel *i*

```
## Get mean parcel signal
    3dmaskave \
        -quiet \
        -mrange $i $i \
        -mask atls_pp_T.nii.gz \
    ${func}_cl_norm_T.nii.gz > _t${i}.1D
```

Preprocessing steps

1. Preprocessing of anatomical images
2. Preprocessing of functional images
3. Anatomical standardization of functional images
4. Removal of noise signal

5. Construction of nodes
6. **Construction of links**

Computation of wavelet correlations

```
wf = 'd4';                              % wavelet filter
s = 3;                                  % wavelet scale
[n t] = size(M);
ts = modwt_num_nonboundary_coef(wf,t,s);
Ms=zeros([n ts]);
for j = 1:n
    ms = modwt(M(j,:), wf, s, 'circular').';
    Ms(j,:) = ms(s,(t+1-ts):t);
end
M_net = corrcoef(Ms.');
```

The final network