## Too long did not read (TL;DR) tutorial.

1. To process your own files, you need to downsample your data. Use **Downsample\_Inscopix() or Downsample\_avi().**
2. To motion correct your files you need to run **MC\_Batch()**;

\*\*\*Demo files are already motion corrected and downsampled.\*\*\*

1. To align demo videos, run **align\_sessions\_PV(10,2.5,1).**
2. To extract neural signal run **CNMFe\_app** and choose appropriate initialization thresholds.
3. Then run **CNMFe\_batch(parin)** to extract the calcium signals.

Optional post-processing

1. Load the .mat file with the processed data.
2. Label false positives with '**ix=postprocessing\_app(neuron);**
3. Run **neuron.viewNeurons(find(ix), neuron.C\_raw);** to check and delete labeled components.
4. Run **manually\_update\_residuals(neuron,use\_parallel,max\_frame)** to pick potentially missed neurons from the residual video.
5. Run **save\_workspace(neuron);** to save results.

## Detailed tutorial:

The following tutorial explains the process of aligning multiple sessions and processing concatenated calcium imaging files. The demo files provided have already undergone motion correction. To process your own calcium imaging data, you must first down-sample and motion correct your files.

You can downsample your videos using the following functions:

• **Downsample\_Inscopix(ds\_f)** for Inscopix videos (.ISXD)

• **Downsample\_avi(ds\_f)** for AVI videos, such as those obtained with UCLA miniscope

Here, ds\_f represents the downsampling factor. We recommend using a 4x factor for Inscopix videos and 2x for miniscope videos. Note that to downsample Inscopix files stored as ‘.ISXD’, you need to first install the Inscopix Data Processing software (currently tested with version 1.6.0). These functions will convert your videos to the '.h5' file format, which is used by CaliAli.

To motion correct your ".h5" videos, you can run **MC\_Batch([], non\_rigid)**;, where non\_rigid is a Boolean variable indicating whether to perform non-rigid motion correction or not. We recommend first monitoring the Ca+2 imaging video with **view\_Ca\_video()** to determine whether non-rigid motion is present or not.

===========Alignment, concatenation, and processing of the DEMO files===============

To align sessions, we use the function ‘**align\_sessions\_PV(sf, gSig, n\_enhanced)**’. Here, "sf" represents the frame rate, "gSig" is calculated by multiplying the average neuron size with 0.25, and "n\_enhanced" is a Boolean variable that determines whether to perform the MIN1pipe background subtraction module or not. We recommend setting "n\_enhanced" to 1 in most cases for optimal results.

1. To process the demo data run: **‘align\_sessions\_PV(10,2.5,1)’.**

This will call a “select files request”:

Graphical user interface

Description automatically generated

1. **Choose the files you want to align. Be sure to add them in order. For the demo files, choose all four files (vs1, vs2, vs3, and vs4). Then press done.**

You should see the following in the command window:

Text, letter

Description automatically generated

If blood vessels are not stable or there are drastic changes in the field of view you may receive a warning message.

After aligning the sessions, a .h5 file with an "\_Aligned" suffix will be created, along with a .mat file that stores relevant variables for subsequent analysis.

\*\*\* NOTE: In case you want to use CaliAli to process individual files, without tracking across multiple sessions, you can run “detrend\_Batch(sf,gSig,neuron\_enhance)”. This will only perform detrending/background subtraction and calculate the necessary projection for subsequent analysis.

Graphical user interface, application

Description automatically generated

The .mat file contains variables that are useful to evaluate alignment performance. Particularly, the variable **P** includes all the projection used to align sessions.

Text

Description automatically generated

1. You can monitor alignment performance by opening vs4\_Aligned.mat and running: “**implay(P.(3)(1,:).(5){1,1})”.**

This will show a projection of the BV and neurons of each session:

Graphical user interface, application

Description automatically generated

1. After confirming that no errors occur during session alignment and concatenation, proceed to extract neural signals. First, it is necessary to estimate suitable initialization thresholds. To do this, execute "CNMFe\_app," which will open the corresponding graphical user interface (GUI):

Graphical user interface, text, application

Description automatically generated

1. Press “Load Data” and chose the “vs4\_Aligned.h5” file. Other files can be chosen to process several videos simultaneously. The GUI should look like this:

Table

Description automatically generated

1. **Press the green “Press me” cell**. Note that the called app may take some seconds to load. Avoid pressing the cell repeatedly to avoid opening several windows.

This will open the following GUI:

Graphical user interface, application

Description automatically generated

Alternatively, you can draw a mask around the regions where neurons are located to prevent the initialization of non-neural signals within the field of view. In this specific demonstration, a mask is not necessary. To create a mask, click the "Draw Mask" button and outline a polygon surrounding the neurons. Ensure that the polygon is drawn over the correlation image (it does not work in other windows).

Graphical user interface

Description automatically generated with medium confidence

1. **Set up the Correlation and PNR threshold as shown in the picture:**

Graphical user interface, application

Description automatically generated

Note that only the seed pixel in the PNR\*Correlation matter for posterior analysis. Press Ok! When finished.

This will update the PNR and Coor parameters in the table:

Graphical user interface, application

Description automatically generated

You can change the frame rate and gSig if you are analyzing a file different from the demo.

1. **Press done!** This will create a new variable “parin” in the workspace.

Graphical user interface, application

Description automatically generated

This variable stores the necessary parameter to run in CNMFe.

1. Initiate neural extraction by running "CNMFe\_batch(parin);". The process may take anywhere from 3 to 15 minutes, depending on your computer's hardware capabilities.
2. After completing the neural extraction, locate the .mat file containing the processed data. You can find this file in the following directory: "Demo\vs4\_Aligned\_source\_extraction\frames\_1\_4000\LOGS\_\*\*\*". Three .mat files will be saved, each representing a different stage of the calcium extraction process. Open the most recent .mat file, which corresponds to the final stage.

This will load the class object “neuron” which holds all the relevant parameters obtained during the neural extraction.

## Alternative post-processing

1. To post process and evaluate the performance of the extracted traces run: **ix=postprocessing\_app(neuron);**

This will show the following GUI:

Square

Description automatically generated

This GUI displays the detected spatial components' contours over the correlation image. You can right-click to zoom in on the image and select neural contours, which will then be plotted in the adjacent window on the right.

Graphical user interface, application

Description automatically generated

You can label false positives by left clicking on the corresponding neural contour. This will change its color to white.

Alternatively, you can use the “Separate Spatial” bouton to identify false positives based on the neuron shapes.

This will bring up the following GUI:

Graphical user interface, application

Description automatically generated

The spatial components here are organized based on their homogeneity in relation to the rest of the population. As a result, the majority of non-neural components can be found towards the end of the list. This is useful to identify dendrites or other structures that do not correspond to neurons:

Graphical user interface, application

Description automatically generated

Right click on suspicious components and press the [ >>] button to separate them from the rest.

Graphical user interface

Description automatically generated

Once you have confirmed the separation of components, press "Done!"

The components stored on the right side of the window will now be displayed with white outlines, indicating that they have been labeled as false positives.

A picture containing graphical user interface

Description automatically generated

You can right click these components to remove the false positive label.

Once you have labeled the false positive neurons press Done!

This will create a variable “ix” with the id of all the false positives components.

You can review and delete these components if necessary by executing the following command: **neuron.viewNeurons(find(ix), neuron.C\_raw);**

Picking missing neurons from the residual signal:

Some neurons may remain unextracted after the initial processing. To extract potentially missed neurons run: **‘manually\_update\_residuals(neuron,use\_parallel,max\_frame)’.** This will bring the following GUI:

Calendar

Description automatically generated with low confidence

The first row displays the three images initially used for neural initialization. The second row presents the residual images after removing the detected component from the video data. These images help in identifying neurons that may not have been extracted.

Carefully examine all three images and click on any neurons that might not have been extracted. Generally, true neural signals can be clearly observed in all three images.

A picture containing calendar

Description automatically generated

Once satisfied, press 'Ok!' This will initialize the newly chosen seed pixel and re-execute the NMF iteration. Be aware that this process may take some time, depending on the size of the video being processed.

1. Once finished, be sure to save results by running ‘**save\_workspace(neuron);’**

**Other useful commands:**

**to visualize temporal traces:**

* view\_traces(neuron);

**To visualize neurons contours:**

* neuron.Coor=[]
* neuron.show\_contours(0.9, [], neuron.PNR, 0); %PNR
* neuron.show\_contours(0.6, [], neuron.Cn,0); %CORR
* neuron.show\_contours(0.6, [], neuron.Cn.\*neuron.PNR,0); %PNR\*CORR

**To save results in a new path run these lines and choose the new folder:**

* neuron.P.log\_file=strcat(uigetdir,filesep,'log\_',date,'.txt');
* neuron.P.log\_folder=strcat(uigetdir,'\'); %update the folder
* cnmfe\_path = neuron.save\_workspace();