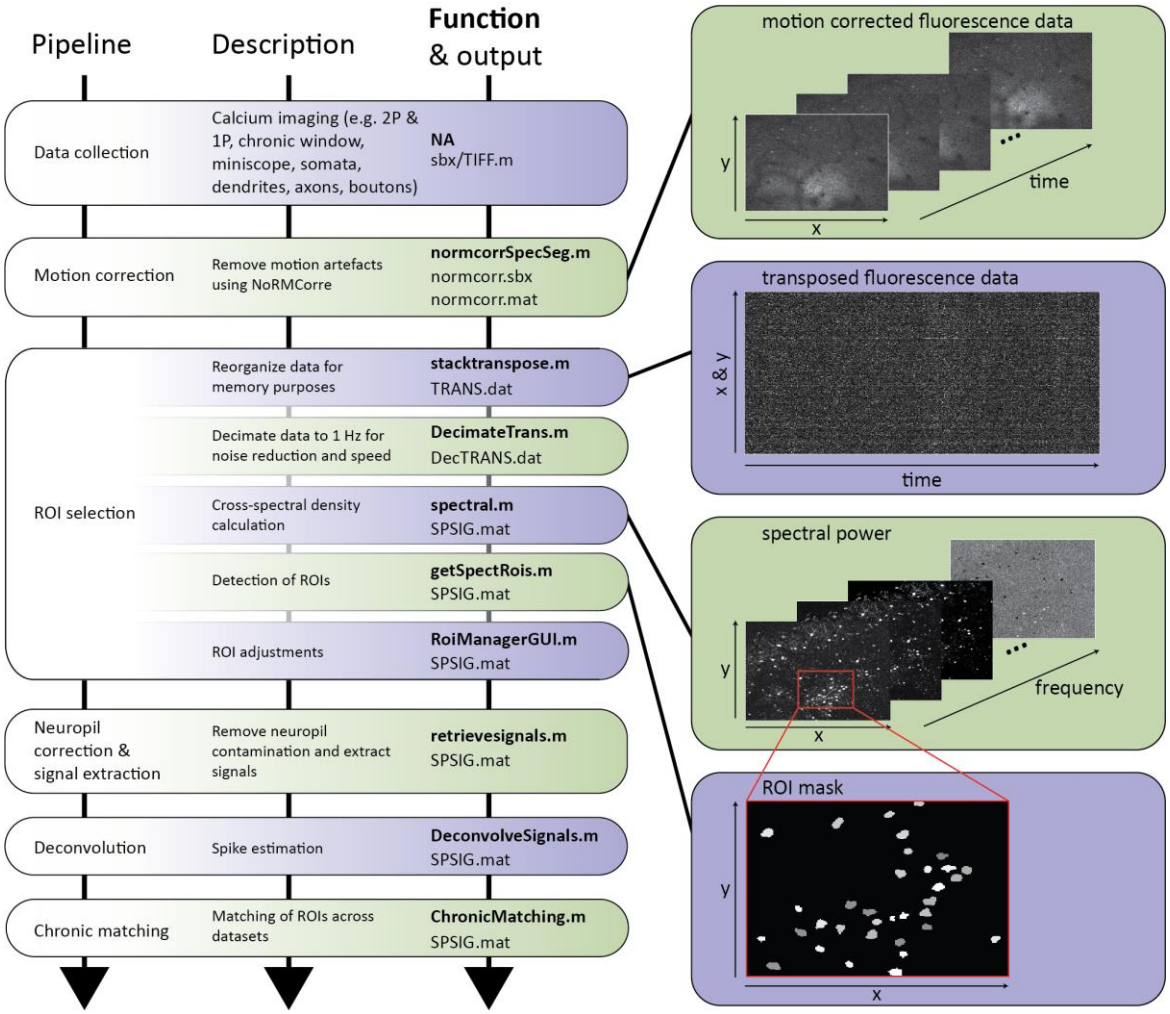
Contents



Introduction

This manual covers practical information on using the Spectral Segmentation toolbox pre-processing pipeline. For more information, see the preprint paper (Kraker et al., 2020).

There are two easy ways to run the pipeline: Via the **SpectralPipeline** or **AutomatedAnalysis** scripts.

The script **SpectralPipeline**.m in the main folder covers all the main functions of the pipeline. All the functions can be called without input, each function will ask for which file to analyse. This script is useful for manually processing the data. Run the script step by step. Settings in most functions/ steps of the pipeline are visible and changeable in the top of that function.
The script **AutomatedAnalysis**.m in the main folder is the script that we use to automatically execute all the pre-processing steps of the analysis for as many .sbx files as you request, until the RoiManagerGUI step, which requires manual input. Some of the steps of the pipeline are optional. The settings of the analysis can be changed either via the code itself, or via input prompts which guide the user through all the standard options.

The pipeline uses a naming convention which it uses to search for files. If you deviate from this naming convention the toolbox will be harder to use.

Transposed files should end their filename with *_Trans.dat*
Decimated transposed files should end their filename with *_DecTrans.dat*
Spectral/ROI files should end their filename with *_SPSIG.mat*

Fluorescence data

Raw data of non-sbx format

The SpecSeg toolbox starts with .sbx files, which is the format that scanbox uses. Sbx files are simply binary format memory files in which the data are ordered in width, height and number of frames that are produced by the scanner, along with an additional metadata file in mat format. We do not have code to perform motion correction on file formats other than sbx files. But NoRMCorre, which is necessary for the motion correction, can handle a lot of file formats. If your raw data does not consist of .sbx files, we recommend to convert your file to sbx format.

Converting Tiff files to sbx data

There are multiple formats in which tiff files are saved. For each format there is a different script to convert the data to sbx format.

Option 1. A single tiff file contains all the frames of the dataset. Use **TiffStack2Sbx**.m. This script can load as many tiff files as you request via the pop-up, converting each selected tiff file to a separate sbx file. Press cancel in the pop-up to stop selecting files.

Option 2. Tiff files which consist of one frame per tiff file, filling an entire folder for one video, can be converted to an sbx file via **TiffImages2Sbx**.m. This script will first ask for the folder in which the tiff images are stored, then it asks for where to save the resulting sbx file and how to call that file.

Option 3. Tiff files have a maximum file size, while sbx files do not. So it can be the case multiple tiff files, containing many frames each, belong to one recording. They can be combined into one sbx dataset using **TiffStack2OneSbx**. This script can load as many tiff files as you request via the pop-up. Press cancel in the pop-up to stop selecting files. Then it asks for where to save the one resulting sbx file and how to call that file.

Converting Hdf5 (H5) files to sbx data

H5 files can be converted with the script **Hdf52Sbx**.m

Raw sbx data

An sbx file can store video of multiple imaged depths by having consecutive frames as different depths before moving on to the next time step. In the SpecSeg pipeline files with multiple depths are split into separate files during the motion correction step. They can also be split without motion correction via the **Showsbx** .m GUI. Every Sbx file has an accompanying .mat file with a struct called *info* that contains information about the data. Some of the standard fields are:

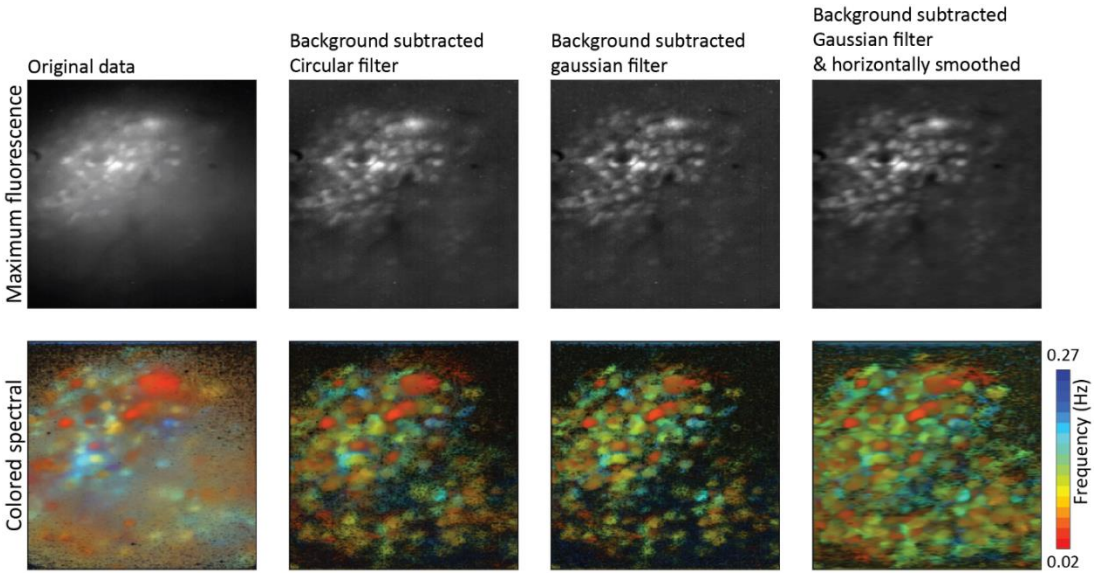
Info.Freq = sampling frequency in Hz.
info.max_idx = how many frames in the file. Frames go through both imaging depth and time.
Info.sz = [height of frame in pixels, width of frame in pixels];
Info.Perm = In which dimensions is which data stored. Meaning of numbers are: 1: width. 2: height. 3: color channels (PMTs). 4 frames & depth.
info.nsamples = the number of bytes for each frame.
info.nchan = how many photon multiplier tubes (PMTs) were used/ How many color channels in data
info.channel = Which PMT was used. 1 means both PMT 0 and 1 were used. 2 = PMT 0. 3 = PMT 1
info.resfreq = frequency of the resonance mirror

info.Slice = which slice/ imaged depth of the data does this file belong to?
info.Slices = how many depths were imaged.

Raw data needs to be motion corrected. We use the toolbox NoRMCorre (Pnevmatikakis and Giovannucci 2017) for motion correction. However, NoRMCorre cannot recognize sbx files. So we have adapted some of their code so NoRMCorre can be used on sbx files. To be able to apply motion correction on sbx files, download NoRMCorre from github: https://github.com/flatironinstitute/NoRMCorre. Don't forget to add NoRMCorre to your MATLAB paths. The motion correction can be applied by calling the GUI **Showsbx**.m, or via the **AutomatedAnalysis** script.

Background fluorescence subtraction

One-photon and miniscope calcium imaging data often suffers from out of focus fluorescence and extreme vignetting. The vignetting obscures neurons that are present at the dark edges of the image. These problem can be mitigated by background subtraction. The background subtraction can be called via the function **BackgroundSubtractSbx**.m, which can also be used as a script.
The background is estimated by convolving with a circular or average filter of 100x100 pixels large. This background image gets subtracted from the original frame. This is done for each frame.
Another form of noise in miniscope data is line read noise from the CMOS sensor, resulting in vertical or horizontal banding. BackgroundSubtractSbx can add Gaussian smoothing in either the vertical or the horizontal direction to decrease that banding. The smoothing step also increases correlation between pixels, which can increase spectral power. Hard to detect neurons become clearer, but patches of signal with non-neuron origin also get higher correlation. The effect of different background subtraction parameters can be tested in **BackgroundSubtractSbxExampleFrame**.m. This script can select multiple timepoints and extract data from certain locations, to get a fast estimation of time traces when changing parameters. The parameters have to be changed by editing them in the script.



Effect of background subtraction in miniscope data. Data was recorded in mouse M2. Top row shows maximum fluorescence projection. The bottom row shows spectral power, with different colors for different frequencies. Many neurons are present in the data, this is hard to see in the original data because of vignetting (see Maximum fluorescence). The spectral power has trouble with the original data because the entire image brightness fluctuates in synchrony with the neurons, causing correlation everywhere in certain frequencies. Background subtraction with a circular filter achieves near identical results as a larger Gaussian filter. This is one reason why the circular filter is preferred; smaller filter is faster.

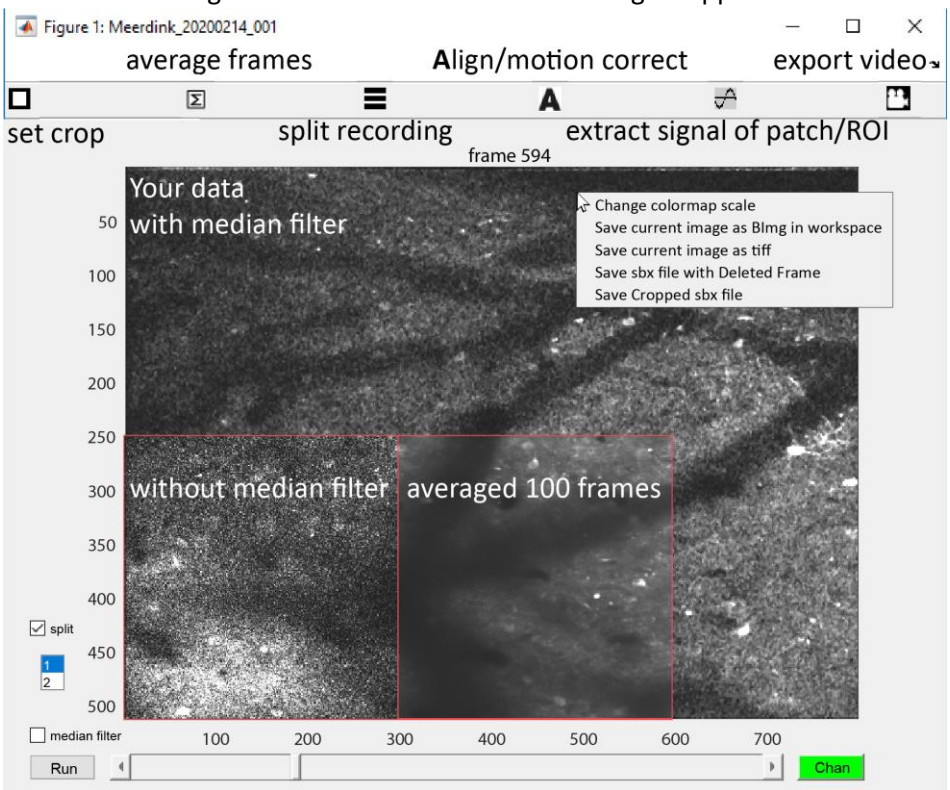
Showsbx.m (to apply motion correction and other things on sbx data)

The Showsbx.m GUI is able to play sbx files and call many functions, including motion correction. This GUI is a nice first step to check your data for any artefacts.

Start the GUI by executing *Showsbx* in the Matlab command in window or by running it via the SpectralPipeline script. Then select which file to open in the pop up menu that appears. If Showsbx can't detect how many depths are imaged in the sbx file, it will ask how many slices (depths) are present in the file.

The contrast of the data can be changed by right clicking in the main image (Change colormap scale).

- The motion correction can be called by pressing the **A** button. The motion corrected file gets saved into the same folder in a new sbx file with '_normcorr' appended to the original name. If the Sbx file consists of multiple depths the motion corrected file will be split into multiple files with '_Splitx_normcorr' appended, with x indicating the split number. If the edges of the recording contain artefacts they can be removed by cropping. The crop will be applied when aligning the data.
- Cropping the data can done by activating the square bordered button 'set crop' (see figure). After the button is activated, set the crop by clicking twice in your data image, once for each corner. Then press enter. A rectangle will appear which denotes what data will be included when aligning your data.
- The button *average frames* allows the plotting of an average fluorescence projection from the current frame until a number of frames later, the standard is 100 frames.
- *Split recording* will split the current sbx file into multiple files, each file consists of a different depth. The number of depths is the number splits, which is determined by info.Slices.
- *Extract signal of patch/ROI* saves the fluorescence signal over time from a patch of data selected after pressing the button.
- *Export video* creates an mp4 of the entire sbx file, with the crop settings applied.
- *Save Cropped sbx file* saves a new sbx file with '_copy' appended to the name, which is a cropped version of the original sbx file. No motion correction gets applied in this case.



The Showsbx GUI. Showing a two-photon recording of 2 depths, imaged in a mouse injected with GCaMP6f in V1 layer 2-3.

ROI selection

Transpose data

The transposed data is necessary for the retrieval of fluorescence traces.
The motion corrected sbx data is transposed with StackTranspose.m.
When calling the function without input it will ask which .sbx file to transpose, followed by where to save the output. StackTranspose uses a mex function. This is C++ code compiled for Matlab. A compiled mex function for 64 bit windows is provided. However, on some Matlab versions or operating systems this mex file may not work. The C++ source code is present in the git. If the Zorder mex file does not work, a version for your system can be compijled via Visual Studio using MATLAB.

Decimate data

To remove noise and decrease the amount of datapoints we often decimate/subsample the data to ~1Hz. Decimating data takes quite some time, so to save time decimated data is saved into a separate file with **DecimateTrans**.m. DecimateTrans.m creates a file with the filename of the original recording, with _DecTrans appended.
The decimated data is necessary for the spectral calculation (spectral.m). It is optional for other functions but can be used by many, speeding them up. These functions first search for the decTrans.dat file before the regular transposed file. Functions that can optionally use the decimated trans files are: RoiManagerGUI, getSpectRois, CalcSpatialCorrFun.

Spectral calculation

Run **spectral**.m to calculate the cross-spectral power. This function creates the SPSIG.mat file.
The spectral calculation requires the decimated transposed data. The data needs to be longer than 60 seconds. The spectral components are stored in the variable *SPic*, the frequencies that correspond to the *SPic* 3rd dimension are stored in *Sax*.

Representative fluorescence images

High quality fluorescence images of your data can be made using the function **FluorescenceImgSbx**.m. It uses the regular motion corrected fluorescence data, not the transposed data.
The projections are made by taking short average projections of multiple frames (chunks) to reduce noise, and using these chunks for a maximum and an average projection. So there are two variables that can be changed in this procedure.
- *nframes*: How many frames are averaged together to use as a picture for the average projection: how many frames in each chunk.
- *nchunks*: How many chunks are used for the average projection.
Higher nframes will result in less noisy images, but very short peaks in fluorescence get averaged out more. So the max projection end result will look slightly more like an average projection.
Higher nchunks means higher quality, because more data is included, but the procedure takes longer. Also, there is no limit so you can take overlapping chunks. This means the same frames can be used more than once in the projection. FluorescenceImgSbx outputs the average fluorescence image *BImgAverage*, the maximum fluorescence projection is saved in *BImgMax*. Information about

how the images were created is stored in the struct *infoImg*, this also contains the percentage of frames used of the recording to create the images in *infoImg*.*percent*.

Get ROIs

getSpectrois.m is used to automatically create ROIs via the cross-spectral power images and parameters specified by the user. These parameters are stored in the variable *spar* (spectral parameters), which also gets saved to the current folder as spar.mat, so getSpectrois can load in the previously used parameters. When calling getSpectrois select the SPSIG file via the pop-up menu, and then set parameters for the ROI selection with the **SpectParArm** UI (see figure below). This UI requires an SPSIG file to show a preview of the data. If you want to set the *spar* variable without loading an SPSIG file you can use the **Spectroiparm**.m UI.
In the SpectParArm UI the top image shows all frequency components, with different colors for different frequencies. It also gives feedback on the selected parameters.

Parameters to set:

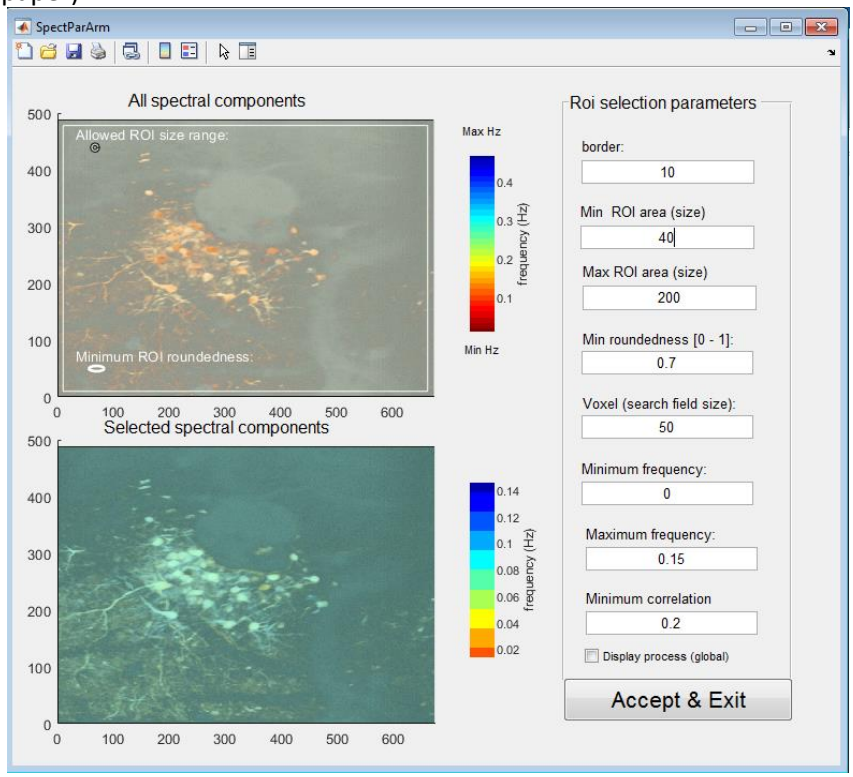
Border: Defines the minimum distance from the edge of the image for finding ROI centres. The selected border is shown with the white rectangle just inside the top image.

Min & Max ROI area (size): The size of ROIs is calculated by how many pixels of the mask/ image they occupy. Feedback is given by a donut (see top left of top image).

Min roundness: the roundness is calculated as the ratio of the area over the square of the perimeter (length of contour) of the ROI. An estimation of the minimum roundness of ROIs to be found is shown in the lower left corner of the top image.

Minimum & maximum frequency select which frequency components to use for the ROI search. This has a large effect on how many ROIs are going to be found. The lower image shows the spectral components that are selected, so you can estimate whether artefacts or neurons are included.

Minimum correlation: The correlation refers to the mean square of the correlations (R^2) of the fluorescence signal from the pixel with the highest spectral power in the ROI with the signal from all other pixels in the ROI. If this value is very low, the ROI signal is mainly uncorrelated noise (see paper).



SpectParArm UI. Select parameters for the ROI selection.

Edit ROIs

ROIs can be edited via the **RoiManagerGUI**, which has a separate manual, located in the doc folder.
https://github.com/Leveltlab/SpectralSegmentation/blob/master/docs/RoiManagerGUI_Manual_V3.pdf

Signal extraction

Fluorescence signal of ROIs

Retrievesignals.m is the script that retrieves the fluorescence signal from ROIs. It uses the regular transposed data, not the decimated data. For the neuropil correction some parameters can be changed. The function outputs the following signals:
sigRaw: raw fluorescence for all the ROIs: the average of all the pixels in each ROI.
sig: Df/f, as calculated with **basecorrect**.m.
sigBack: The Df/f of the neuropil ROIs.
sigCorrected: Df/f with neuropil subtraction before converting to Df/f
Basecorrect calculates the Df/f by estimating the baseline fluorescence using a 10th percentile filter of 5½ minutes wide. This creates a changing baseline, allowing for very slow changes in the fluorescence, while excluding faster spikes.

Spike estimation with MLspike

DeconvolveSignals.m uses MLspike (Deneux et al., 2016) for spike estimation of the retrieved fluorescence traces. To be able to use it download the code from brick and spike at https://github.com/MLspike, and add it to the MATLAB path. The spike estimates are saved in the same format as the fluorescence signal. The outputted signals are:
Decon: Spiking data of the Df/f from *sig*.
Den: Denoised signal of the Df/f from *sig*.
DeconCorrected: Spiking data of the neuropil corrected Df/f *sigBack.*
DenCorrected: Denoised signal of the neuropil corrected Df/f *sigBack*.

Chronic matching

Jumping into chronic without creating ROIs with spectral pipeline.

The chronic matching provided in the toolbox requires some specific variables that are created in the Spectral processing pipeline. However, you can use ROIs that you have found with other toolboxes. To be able to execute the Chronic matching as provided, you need to have pre-processed the data until and including the spectral calculation step.
The essential variables for chronic matching are:

Mask (2D double [height x width]): An image that tells where ROIs are. Each pixel corresponds to an ROI by the number it has. 0 means no ROI present. Usually ROIs do not consist of multiple areas; they touch themselves everywhere. This variable is completely essential, and you need to provide this one yourself.

PP (struct): This struct has a lot of info about the ROIs in Mask, like size (PP.A), Number of ROIs (PP.Cnt), contour coordinates (PP.Con), ROI seedpoint location (PP.P) etc. If you have a mask, those properties can be created using **PPfromMaskFile**.m. Then the rest of the ROI properties can be retrieved by **PPModernize.m**. PPModernize requires transposed data. For chronic matching, the only required data of PP are the variables mentioned above, so it is not essential to run PPModernize.

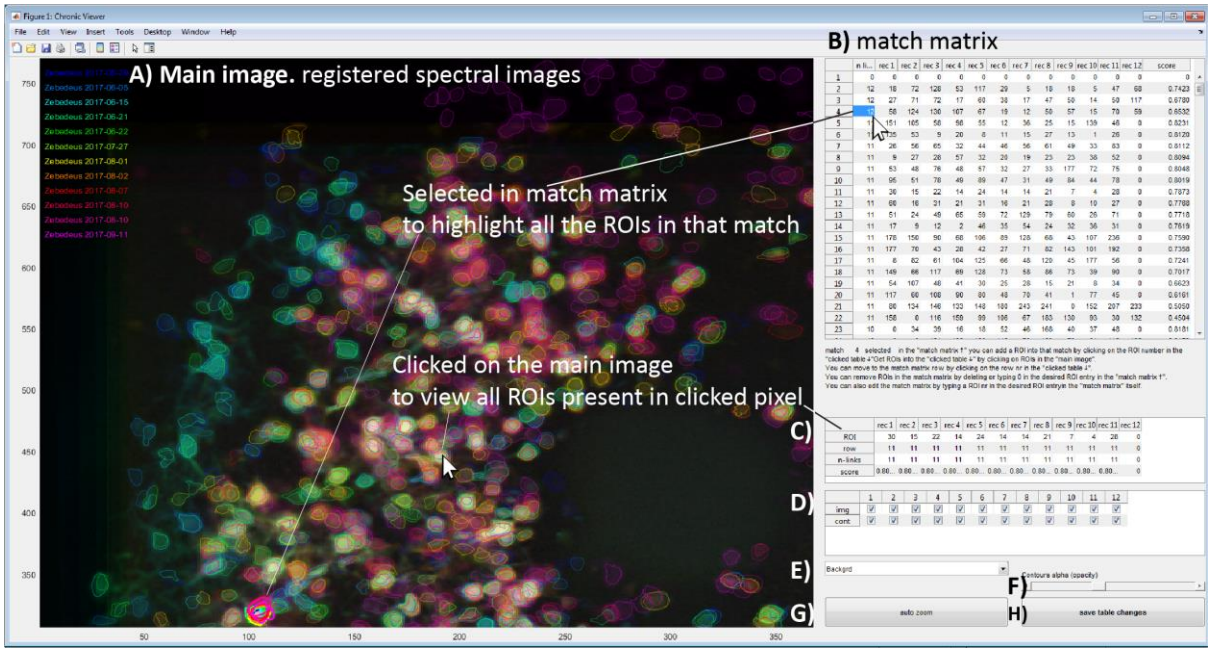
BImg or *BImgMax* or *BImgAverage* (2D doubles [height x width]): Images for registration.

Chronic matching

For chronic matching, execute the **ChronicMatching** script step by step. When the chronic matching is complete, the last step is to check the matches with the **ChronicViewer** UI (see figure). In this example there are 12 recording sessions, recorded over a period of 105 days in mouse V1 layer 2/3 expressing GCaMP6f. The UI displays the registered spectral images in the main image **A**. The visible neuron population shifted over time, in the early recordings (blue, cyan, green) a lot of neurons are visible on the left side. These were not visible in other recordings. The last recording had the largest differences. Perhaps that time a different depth was imaged accidentally. **B** contains the match matrix (via a MATLAB UI table) which shows the chronically matched ROIs, one match per row. The left column says how many ROIs are present in that match. The right column contains the overlap scores, which is the average of the fraction of ROI pixel overlap between all the different sessions for this match. The other columns show the identification number of the matched ROIs for each recording.

If you wonder where ROIs are located in the spectral image, you can click on the ROI ID in the match matrix. This will highlight contours of the corresponding ROIs in the main image. The reverse can also be done. If you want to check whether ROIs are matched together, or simply which ROI ID number belongs to a certain ROI in the image, you can click in the main image. Information about the ROIs you clicked on will be shown in **C**, the "clicked table". To quickly navigate to the match, click on that row nr in row 2.

The match matrix can be edited. To replace or add a ROI, First click on one match in the match matrix. Selection of that match will be confirmed in the text below the match matrix. For example, match 4 is selected in the example image. Secondly, click on a ROI nr in **C**. This will add the clicked ROI into the selected match. The match matrix can also be edited by typing ROI ID numbers into the match matrix/ UI table. Once a ROI is placed in a match, the UI will check whether that ROI is present in another match, and remove it from that match, also recalculating that match's properties. The top row of the match matrix is always empty, so new matches can be added. ROIs can be removed from a match by deleting them directly from the match matrix. **D)** Select which recording session image or ROI contours to show in the main image. **E)** Select other views for the main image. **F)** Contour alpha changes the opacity of the ROI contours. **G) '**auto zoom' toggle enables a zoom to the ROI you clicked on in the match matrix. **H)** Save an edited match table to the MATLAB workspace.



References

Deneux, T., Kaszas, A., Szalay, G., Katona, G., Lakner, T., Grinvald, A., Rózsa, B. & Vanzetta, I. (2016) Accurate spike estimation from noisy calcium signals for ultrafast three-dimensional imaging of large neuronal populations in vivo. *Nature Communications* DOI: 10.1038/ncomms12190

Kraker L., Seignette, K., Thamizharasu, P., Pica, I.F., Levelt, C.N. & Togt, C. (2020) Specseg: cross spectral power-based segmentation of neurons and neurites in chronic calcium imaging datasets. *Biorxiv* doi: 10.1101/2020.10.20.345371

Pnevmatikakis, E. A., & Giovannucci, A. (2017). NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data. *Journal of neuroscience methods*, 291, 83-94. 10.1016/j.jneumeth.2017.07.031