P2 Pipeline

Note: DataFolder refers to the folder with raw data, SaveFolder refers to the folder with all the processed data. Often, you load things from the SaveFolder which can be confusing. Next time, make sure to name it RawFolder and DataFolder.

Inhoud

[GCaMP 3](#_Toc157591931)

[Preprocessing\_Pipeline\_CaCl\_GCaMP 3](#_Toc157591932)

[CompareMovementRecordings 3](#_Toc157591933)

[Transfer Recordings 3](#_Toc157591934)

[ImagesClassification (Labeo) 3](#_Toc157591935)

[MarkMovedFrames 4](#_Toc157591936)

[CorrectDichroic 4](#_Toc157591937)

[Flip Left-Right 4](#_Toc157591938)

[ROI creating + mask 5](#_Toc157591939)

[ClusterRois 5](#_Toc157591940)

[Remove Fluo 475 files 6](#_Toc157591941)

[Coregistration 6](#_Toc157591942)

[Umit\_Pipeline\_CaCl\_GCaMP 7](#_Toc157591943)

[Hemocorrection (Labeo) 7](#_Toc157591944)

[Normalisation and filtering (Labeo) 7](#_Toc157591945)

[HbO HbR (Labeo) 7](#_Toc157591946)

[Pipeline\_CaCl\_GCaMP 8](#_Toc157591947)

[GetTimecourses 8](#_Toc157591948)

[GroupVariables 8](#_Toc157591949)

[CombinedCorrMatrix 8](#_Toc157591950)

[SingleSubjectSPCM 8](#_Toc157591951)

[CombinedSPCM 9](#_Toc157591952)

[SingleSubjectSeedSpread 9](#_Toc157591953)

[CombinedSeedSpread 9](#_Toc157591954)

[Single Subject Plots 10](#_Toc157591955)

[ExampleData\_GCaMP\_HbO\_HbR 10](#_Toc157591956)

[Speckle 11](#_Toc157591957)

[Preprocessing\_Pipeline\_speckle 11](#_Toc157591958)

[CompareMovementRecordings 11](#_Toc157591959)

[Transfer Recordings 11](#_Toc157591960)

[PlotTreadmill 11](#_Toc157591961)

[ImagesClassification 11](#_Toc157591962)

[MarkMovedFrames 11](#_Toc157591963)

[Flip Left-Right 11](#_Toc157591964)

[Ultrasound 12](#_Toc157591965)

[Us\_analyis.mlapp (Fred, updated by Marleen) 12](#_Toc157591966)

[M Mode 12](#_Toc157591967)

[PW Mode 12](#_Toc157591968)

[Save 12](#_Toc157591969)

[BugPatchMultiplyMouse 13](#_Toc157591970)

[US\_Analysis\_Pipeline 13](#_Toc157591971)

[US\_analysis\_by\_hand 13](#_Toc157591972)

[PlotUSResultsBoxplot 13](#_Toc157591973)

[PlotUSResultsScatter 14](#_Toc157591974)

# GCaMP

Analysis done on lab computer (Linux)

## Preprocessing\_Pipeline\_CaCl\_GCaMP

Things in the preprocessing pipeline generally do not have a check for if a function is already done within the function. Instead, it is in the pipeline itself, and it checks the AnaReg. Later, in the Pipeline\_CaCl\_GCaMP, the check is usually within the functions, and can be chosen to be overwritten.

### CompareMovementRecordings

Input is the general directory with your mice, which is structured like GCaMP(/Speckle) – Mxx – Mxx-Ax-Rx. First makes a list of all acquisitions in the give folder. Then makes the RecordingOverview(\_Speckle).mat file or opens and modifies it. Creates a table with all mice and their first, second and third acquisition (A1, A2 and A3). Loops through mice, checks per mouse, per acquisition, if there is already a best acquisition found and written in the table. If not, it will check if we already plotted the movement and saved the tif file. If so, it will display that, if not, it will use PlotTreadmill.m to plot the movement. PlotTreadmill.m takes the ai files of the acquisition, looks at the 5th channel (the treadmill information) and plots it. Also looks between 1.63 and 1.65 and takes anything out of those lines as movement. Displays the amount of movement based on that (this is an estimate since the cutoffs differ per recording). Gives a plot. CompareMovementRecordings gives a prompt to click on the best acquisition for that mouse.

Needs:

* DataFolder (GCaMP(/Speckle) folder with all acquisitions and mice)
* PlotTreadmill.m
  + Ai\_\*.bin
* MakeAcqList.m

Gives:

* RecordingOverview(\_Speckle).mat
* Movement-Mxx-Mxx-Ax-Rx.tif in mouse folder

### Transfer Recordings

If we want to use the umIToolbox later on, we need to give the folders for the raw data and the folder where we want to save stuff. To make sure we only take the folders that we need, we will move any recordings that were not the best and put them in a separate folder (GCaMP(/Speckle)\_Non\_Used instead of GCaMP(/Speckle)).

Loops through mice. Takes all the folders that are under one mouse and sees if they exist in the best recordings as noted in the RecordingOverview(\_Speckle). Any folder that is not in RecordingOverview(\_Speckle) will be moved to GCaMP(/Speckle)\_Non\_Used.

Needs:

* RecordingOverview(\_Speckle).mat

Gives:

* Different folder structure

### ImagesClassification (Labeo)

Takes the raw data and makes .dat files that you can work with. Separates the different channels.

Needs:

* Ai\_\*.bin
* Img\_\*.bin
* DataFolder (one acquisition)
* SaveFolder
* BinningSpatial (1)
* BinningTemp (1)
* B\_SubROI (0)

Gives:

* Green.dat
* Red.dat
* Fluo\_567.dat
* Fluo\_475.dat
* speckle.dat
* Green.mat
* Red.mat
* Fluo\_567.mat
* Fluo\_475.mat
* speckle.mat

### MarkMovedFrames

Takes the recorded treadmill data, which has a much more precise sampling time than the acquisition, and puts it into the same number of frames as the acquisition. It takes a cutoff and labels everything above that as movement. It has a buffer of 10 frames for each frame that had recorded movement.

Needs:

* DataFolder
* SaveFolder
* Fluo\_567.mat
* speckle.mat

Gives:

* MovMask.mat

### CorrectDichroic

The way the current set-up was done requires a re-aligning of the frames. Camera 1 and camera 2 differ a couple of mm in their field of view. **This needs to be done after ImageClassification but before any calculation regarding fluo or HbO/HbR**. It loads a single frame from red.dat and green.dat, increases the small characteristics in them, and compares them to each other. The moving picture is the green one and the fluo one (both on cam1).

Needs:

* Red.dat, green.dat, fluo\_567.dat

Gives:

* Red.dat, green.dat, fluo\_567.dat

### Flip Left-Right

The images of the set-up are mirrored, where left is right and right is left. (This is double checked by recordings made with a piece of paper where L and R indicate left and right. Both Speckle and GCaMP images in our setup are flipped.) To fix this, we simply grab the .dat files and do a fliplr and safe them again. We do this for green, red and fluo\_567/speckle, and save these events separately in the anaReg.

Needs:

* Red.dat, green.dat, fluo\_567.dat/speckle.dat

Gives:

* Red.dat, green.dat, fluo\_567.dat/speckle.dat

### ROI creating + mask

In order to align the frames with the toolbox, we would have to save a reference frame. We do this at the same time as registering the Allen atlas on the mouse brain. It opens the ROImanager with a fluo image, and also shows a green image in a figure. In ROImanager, do the following steps:

* Image – Set origin – New – drag to bregma, right mouse-click to set
* Image – Set origin – Align image to origin – drag to lambda, right mouse-click to set and correct for tilt in frame
* Image – Set pixel size – 50 pixels per mm
* Image – Mask – Draw new – Draw the mask by clicking points, you can still drag the points after setting them, double click inside the mask to confirm
* Image – Image reference file… - Export – save as ImagingReferenceFrame.mat in the mouse folder
* Create – Mouse Allen Brain Atlas – Areas – Select areas – close window to select them all – drag atlas to what seems to be fitting – double click inside atlas to confirm
* File – Save as… - ROImasks\_data.mat

Needs:

* Fluo\_567.dat
* Green.dat

Gives:

* ImagingReferenceFrame.mat
* ROImasks\_data.mat

### ClusterRois

Takes the smaller regions as made with the ROImanager and saves in ROImasks\_data.mat. The definition of which smaller regions belong in which clusters is saved, but is made with the commented code below the function if it ever needs to be redone. This function saves in BigROI.mat, where it has the variables BigROI (all of the regions of interest separately), and AtlasMask. AtlasMask is the most important one, it is all of the regions with different numbers as a mask for that mouse. It will be used in GetTimecourses.

Needs:

* ROImasks\_data.mat
* PooledROI.mat

Gives:

* BigROI.mat, with variables BigROI, AtlasMask, regions

### Remove Fluo 475 files

Since our system is slightly Frankenstein-ish, we have recorded on the fluo\_475 channel as well, even though this data is nonsense. This is an automated way to delete those files and save space on the computer. Since it’s deleting files, an askbox will ask if you’re sure.

### Coregistration

We only fit the ROI once per mouse, and save this file in the general folder of that mouse. This means that we will have to make sure that the brain is at the same place in the image for all the acquisitions of the same mouse. Coregistration requires a manual check, to see if it was done correctly. Therefore, this function will only be done if the ManualInput is set to 1. It corrects based on the green channel but this is modifiable.

Needs:

* Fluo\_567.dat
* Red.dat
* Green.dat

Gives:

* Fluo\_567.dat (shifted to fit A1)
* Red.dat (shifted to fit A1)
* Green.dat (shifted to fit A1)

## Umit\_Pipeline\_CaCl\_GCaMP

Does what the Umit toolbox was supposed to do.

### Hemocorrection (Labeo)

Calls Labeo function for hemodynamic correction. Right now (1-9-2023), it is necessary to load the fluo data before calling the function. This will be fixed later probably. Before it saves, it changes all the nans to zeros, in order to do the normalization later. The .mat file is made by a separate function (Make\_HemoCorr\_Matfile) and contains the same information as the fluo\_567.mat file, except the datFile is different.

Needs:

* Fluo\_567.dat
* Fluo\_567.mat

Gives:

* hemoCorr\_fluo.dat
* hemoCorr\_fluo.mat

### Normalisation and filtering (Labeo)

Calls Labeo function NormalisationFiltering. Filters between 0.3 and 3 Hz in this pipeline.

Needs:

* hemoCorr\_fluo.dat
* hemoCorr\_fluo.mat

Gives:

* hemoCorr\_fluo.dat

### HbO HbR (Labeo)

Calls Labeo function HemoCompute.

Needs:

* green.dat
* red.dat

Gives:

* HbO.dat
* HbR.dat
* HbO.mat
* HbR.mat

## Pipeline\_CaCl\_GCaMP

### GetTimecourses

**Important: GetTimecourses works with imerode to get the center of the ROI. This means that for the retrosplenial cortex, it has three “center points”. This is different from the calculation of the seed for the SPCM and the seedspread.** Calculates the timecourses of the regions of interest. Relies on BigROI.mat. You can choose here if you want to calculate the timecourse of the average of the whole region, or to take a seed in the middle of it (option ‘centroids’ or ‘average’). You specify the datatype with dataname.

Needs:

* BigROI.mat
* Fluo.dat or hbo.dat or variations of that

Gives:

* Timecourses\_xxx.mat

### GroupVariables

Needs a specification of what groups you want. So far (24-10-23) this can be Sex, Group, or {‘Sex’, ‘Group’}, which will make a combination of those. It makes a separate column in the RecordingOverview table if it is a combination (RecordingOverview.Combi). This is not completely necessary anymore (29-11-23) because the CompareMovementRecordings makes the RecordingOverview including a .Combi column. It gives all the possible groups and a code unique for that group that will help to not replicate the plots.

Needs:

* RecordingOverview
* Grouping

Gives:

* (RecordingOverview.Combi, if applicable)
* Groups (names and codes)

### CombinedCorrMatrix

Takes all timecourses that were made with GetTimecourses.

### SingleSubjectSPCM

Gives the Seed Pixel Correlation Map (SPCM) for a single mouse. Since the allen atlas would give way too many maps, it automatically takes the BigROI. It does not check yet whether it was already done (22-8-2023). It takes the data, calculates the centroid of the ROI, and sees how much every other pixel of that mouse correlates with the centroid. It depicts that in an image, together with two lines to show the pixel to mm ratio. This is hardcoded at 50 pix per mm.

Needs:

* BigROI.mat
* Mov\_aligned.dat or fluo.dat

Gives:

* SPCM for all 10 ROIs in BigROI, saved in SPCM/Individual

### CombinedSPCM

You can choose on which data you make this function run. Loads the recordingoverview, the BigROI reference mask. Checks which mice are in which groups. Goes per group (nacl/cacl), per mouse. Takes the BigROI mask per mouse, and aligns it to the reference mask so that all brains are located on the same area when combining them later. Opens data, optionally does a GSR. Go per region of interest, take the middle of it (based on the reference atlas, not the individual mouse atlas, so that the cropping of the borders does not make a difference). Correlate middle of the ROI with every other pixel in the brain, and save it in AllSPCM. When this is done for all regions, for all mice in a single group, take the average of all the mice. Do the same for the second group. Save the seed pixel correlation maps.

Needs:

* Recordingoverview.mat
* BigROI\_ReferenceMask.mat
* BigROI.mat
* .dat file of choice

Gives:

* SPCM of NaCl and CaCl of all mice combined

### SingleSubjectSeedSpread

This is to see how big the correlation area of the seed is. Is the correlation of the seed high but only in a very small area around the seed, or does it spread much further? Goes per mouse. Loads the needed data and makes a CorrSpread variable with the size of the number of regions (10) and a size that we estimate, in this case we put 15. This means that we do 15 steps of calculating the correlation of the periphery with the seed.

We start with the seed. We make it expand with 5 pixels, and then we subtract the previous seed+periphery, so that we only have a 5 pixel wide line as a circle. We take all the pixels in that circle, and calculate the correlation with the seed for each pixel. We take the average of that and save it in CorrSpread. This way, you get a maximum of 15 values in CorrSpread, with each value being of a circle that is bigger than the previous, and thus further from the seed. If the circle is so far out of the ROI that we have less than 11 pixels, we stop calculating.

Needs:

* BigROI.mat (the AtlasMask and regions)
* Fluo.dat or HbO.dat or hemoCorr\_fluo.dat

Gives:

### CombinedSeedSpread

Needs:

Gives:

## Single Subject Plots

### ExampleData\_GCaMP\_HbO\_HbR

This takes a recording and plots the GCaMP data on one axis, and the HbO/HbR data on the other axis. You can also plot several ones in the same image, if you uncomment lines 79-93 and if you comment the ylabel lines in the function PlotExampleData below.

Needs:

* DataFolder
* Coords and timing (takes pixel 300,300 and timing 800 – 1100 frames by default but **the best pixel and timing differ per mouse and acquisition** obviously).

Gives:

* Plot with example data

# Speckle

Analysis done on lab computer (Linux)

## Preprocessing\_Pipeline\_speckle

Note: Most of these steps are the same as GCaMP and use the same functions. However, speckle does **not** have an extra folder in the mouse acquisition that is called CtxImg, whereas GCaMP does. Also, speckle has a different frequency. These things are coded into the functions.

### CompareMovementRecordings

See GCaMP.

### Transfer Recordings

See GCaMP.

### PlotTreadmill

See GCaMP.

### ImagesClassification

See GCaMP.

### MarkMovedFrames

See GCaMP.

### Flip Left-Right

See GCaMP.

# Ultrasound

Analysis done on laptop (Windows)

## Us\_analyis.mlapp (Fred, updated by Marleen)

In Project Data Setup: load project file if you already have one. If this gives a multiplication of mice, see BugPatchMultiplyMouse. Check per mouse what the best recording is before starting and mark them so they are easy to identify. Open the ultrasound analysis software by typing us\_analysis in matlab. Add the M Mode and PW Mode for the first mouse. Make sure they are selected in the GUI, and give the mouse a short name. Add the mouse to the dataset by clicking “Associate and Name Current Selection”. Do this for all the mice (or a first selection). Go to the Analysis tab.

#### M Mode

* Take a frame that has data. The ECG Top and ECG Low are lines between which the heartbeat and respiration lines should be. If this is not the case with the standard values, adjust them. It is important that the breathing line is also within the red lines, because it can mess up the estimation of the ECG peaks if this is not the case.
* The “view size” is in mm, and can be found on the scale bar next to the imaged vessel (not the picture, but the bottom part right above the heartbeat). This is often something like 11.0 to 9.2, so the view size is 1.8 (the default). It’s good habit to double check though.
* Take the loop above the MMode Video and zoom in on the vessel on an area that is uninterrupted, in both the vessel and the ECG. Make sure it’s not during a respiration of the mouse of possible. The top and bottom of the vessel have to be visible. Click on “Segment Diameter”. You’ll see a curve coming up. Click on “Estimate ECG Peaks”. This will give lines where the estimated heartbeat was. If you’re happy with this, click on “Add Frame to Data” and “Compute Mean Curve”. If you’re not happy with this, you can change frames and try again, or change the recording with the “Change Recording” button.

#### PW Mode

* Take a frame that has data. The ECG Top and ECG Low are lines between which the heartbeat and respiration lines should be. If this is not the case with the standard values, adjust them. It is important that the breathing line is also within the red lines, because it can mess up the estimation of the ECG peaks if this is not the case.
* The “Speed range” is in mm/s and can be found in the same place as the view size in MMode. The scalebar often goes from plus to minus, for example -392 to 168. The speed range would then be 168+392 = 560.
* Take the loop above the PWV Video and zoom in on the vessel on an area that is uninterrupted, in both the vessel and the ECG. Make sure it’s not during a respiration of the mouse of possible. Sometimes the peak is “cut off” at the bottom and “continues” at the top. You can just take the whole thing in your segmentation. Click on “Segment Diameter”. You’ll see a curve coming up. Click on “Estimate ECG Peaks”. This will give lines where the estimated heartbeat was. If you’re happy with this, click on “Add Frame to Data” and “Compute Mean Curve”. If you’re not happy with this, you can change frames and try again, or change the recording with the “Change Recording” button.

#### Save

* To keep the values of the current mouse, click on “keep mouse data”. This also turns the mouse green in the “Mice” list.
* If you did all the mice, you can click on “Generate CSV File”. If you are missing values for certain mice/acquisitions, the code will give you a pop-up warning you about this.
* If you want it to work with the rest of the code, save it as ResultsUS.

### BugPatchMultiplyMouse

Sometimes, if you load a project file, it will duplicate certain mice. I haven’t found the cause for this yet. If you see that this happened, close the app. Open the BugPatchMultiplyMouse. Load your project file. Run the first lines (1-20). Save the project file.

## US\_Analysis\_Pipeline

This code shows the steps that you have to do for the ultrasound analysis. First, do the us\_analysis application. You can then load the table here. Name the table ResultsUS. This table is very basic so we want to add some things, like the mouse and the side that the ultrasound was done on. We also put a column labeled X. This is to make the spacing of the scatterplot work nicely. If there was trouble for some mice in the app, you can fix this by doing the analysis by hand. After you’ve done that, you can plot the results.

Needs:

* ResultsUS.mat, made in us\_analysis
* Mice.mat file, with columns with ‘CodeOfMouse’, ‘CaClSham’ and ‘MaleFemale’, corresponding to the mice that you did.

Gives:

* Updated ResultsUS.mat file
* Plots for the us results

### US\_analysis\_by\_hand

To do certain acquisitions “by hand”. Give the path of the file you want to analyse, and call it pwv\_file or mmode\_file, depending on the acquisition. Then you can do PW or MMode by either calling US\_analysis\_by\_hand\_PW or US\_analysis\_by\_hand\_MMode with the filename.

Mostly, the script will prompt you with what to do, but it can be that there is a “pause” in the script, where you have to do something and press any key in the command window afterwards. For example, there is code in there that gives you a different cutoff for detecting the upper and lower part of the vessel. This necessitates you to input both cutoffs and the division line that you want (somewhere in the middle of the vessel). In order to do that, an image will be given, and you can check the values. You will then have to press a key in the command window, and it will ask you the cutoff values.

There’s also code to add it to the table you made in the app or with the pipeline. It’s good practice to write down the outcome of the analysis in this script, so that if you lose something you don’t have to redo it.

Don’t forget to save the new results!

Needs:

* Video that you want to analyse

Gives:

* Either mean\_velocity, PI and RI if you do PW, or diameter\_change and pct\_diameter\_change if you do MMode.
* Updated ResultsUS.mat

### PlotUSResultsBoxplot

Takes the ResultsUS and plots them in boxplots with two groups: CaCl and Sham. You can choose to plot all US results (including a example frame of PW data, but not of MMode data) or a “Selection”, which in this case is only PulsatilityIndex and Velocity and an example frame of the PW data.

**This function also calculates statistics.** It gets the p-values for the difference between the groups for each US measurement (t-test). It first checks if the data is normally distributed (Anderson darling test) and if the variance is homogenic (vartest2). If these assumptions are met, it will give a p-value and displays it under the plot. It does not correct for multiple comparisons.

Needs:

* ResultsUS.mat, made in us\_analysis
* Mmread package to get frames of recording

Gives:

* Boxplots of ultrasound results
* P-values

### PlotUSResultsScatter

Takes US results and plots both the left and the right side, connected with a dotted line if the mouse is the same. Does not do statistics. Plots all the results of the US.