**EXTRACTION PIPELINE (data in this repository is already extracted as .mat files)**

Sbatch script

Mat script

Python script

* Make Excel spreadsheet with all information about each recording:

Day: day number in real time (including non-recoridng days), starting at 1

day\_idx: recording day number (excluding non-recording days) starting at 1

date: date in format MM\_DD\_YYYY

animal: animal name

session: session names (e.g. CYL vs RCT)

training: training name (e.g. HAB1, HAB2, TRA1...)

analyze: 1 = analyze, 0 or empty: do not analyze

videoname: the name of the video fiel created by the miniscope software. Needs to be precisely matched.

Trackername: name of corresponding .dat file (position tracker files)

Arena: same as Trackername for arena frame (if exists/needed, otherwise leave blank)

* Copy from Google Drive to Prince Server
* Check with Excel that all files have matching names

While code is supposed to order files correctly, I add 0’s to H/M/S time for single digits ensures it further

Remove useless files

* Folder organization on HPC cluster:

HPC cluster has a home and a scratch folders.

All code files are stored in home. I use scratch for all data (original, in-process and processed). Organization of files in home matters very little. Keeping files very well organized on scratch is very important for easily running standardized code.

I organized as follows: MNS\_DATA>experiement>animal>day>recordings

* Use TiffConversion.sh to convert tiff to avi

Converts to tiff using conversion\_to\_tif\_stack\_singlestack\_wholeday.m or conversion\_to\_tif\_stack\_singlestack\_wholeday\_fcn.m (to run multiple days with SBatch)

**Each folder, each corresponds to a recording and contains multiple avi files, is converted to a tiff file (e.g., for CS expt, 6 folders/day (HC, Cyl, Rec, Cyl, Rec, HC) yield 6 tiff files/day)**

* Motion correction and subsampling

Use MotionCorr.sh to both Motion Correct the tiff files, using MotionCorr\_fcn, and resave them subsampled in one large tiff, using resave\_tiff\_fcn (output file: \*\_Out\_DS\_MCnr.tif).

**Each of tiff file is motion-corrected separately but using the same template: the one created when doing motion correction on the first file of that day. There is then only 1 template per day. Then all the tiff files from a given day are saved, subsampled, in one tif file, resulting in one tif file per day. The resampling procedure computes the average of every 3 frames.**

MotionCorr.sh also runs resave\_tiff\_fcn\_single which saves only the first tif file of the day. This file is then used setting the PNR for the day. Alternatively, you can use the sbatch file ResaveTiffSingleTest.sh to only run resave\_tiff\_fcn\_single.

* Estimate peak to noise value (PNR)

Download output of resave\_tiff\_fcn\_single (\*\_Out\_DS\_MCnr\_testFile.tif) from cluster to Data>Expt>PNR> tifFiles. Then use Save\_PNR to anlyze it. This outputs PNR and correlations in Data>Expt>PNR>out in a .fig and .mat. You can open the .fig or load the mat and plot it yourself. Use the PNR and Correlation data to find an appropriate PNR threshold to seed the CNMF-E algorithm. This is stored in an excel spreadsheet: Cnmfe\_PNR.xlsx. I also stored the cumulative % data point below threshold for record (can be found in the .fig)

**Each day a PNR threshold is chosen separately.**

* Run CNMF-E

Use cnmfe\_script.sh which runs fcn\_batch\_1p, inputing in the .sh file: the expt name, the animal name, the date and the chosen PNR threshold.

**Each daily tif file is analyzed separately using the cnmfe algorithm**

Output files of fcn\_batch\_1p are mat file: format: \*animal\*\_\*date\*\_out\_1b\_ss3.mat

Download \_ss3.mat files to analysis folder: Data>Expt>mat

Download \_template.mat files to analysis folder: Data>Expt>templates. These files contains the motion correction template for each day, they are the output of MotionCorr\_fcn (more specifically of MotionCorr\_server). But they need to be renamed with animal name and day (to \*animal\*\_\*day\*\_template) before being dowloaded. These files are important for cell registration.

Organize and export miniscope timestamps to Data>Expt>timestamps. Organized in folder by day and then named: \*day\*\_timestamp.dat

Organize and export tracker files to Data>Expt>tracker\_files. Organized in folder by day but name do not matter (as long as it matches exactly the name in the excel spreadsheet)

(there was a mistake in the roi function of the cnmf-e package, make sure to use mine if downloading directly from github)

Evaluate best window parameter with: test\_registration\_parameters.py

Then use saveRegistrationByPairs.py to create the registration file using the best parameters

TO CHECK DATA:

* Evaluate fluorescence:

Use CheckFluo, which runs checkFluo\_fcn to extract fluorescence during each recording (\*expt\*\_\*animal\*\_\*day\*\_allVideo.png) and the average fluo across recordings ((\*expt\*\_\*animal\*\_meanVideo.png)

* Evaluate registration

CheckRegistration.sh uses checkRegistration\_fcn to create an avi file from each recording

You can also download a mat file and visualize it or create a video.

* Evaluate alignment:

Use checkAligned to evaluate Evaluate alignment creates \*animal\*\_checkAligned.tiff files which contains the position of imaging across days. Days with large shift are not aligned well and should be excluded.

**ANALYSIS PIPELINE**

Folder organization for analysis

**Python** (python analysis code)

**Matlab** (matlab analysis code)

**Figures** (figures and spreadsheets saved from analysis)

> Experiment (e.g., CircleSquare)

> savedFigs (saved directly from code)

**Data** (data)

> Experiment (e.g., CircleSquare)

> mat (output .mat files (\*\_ss3) from the cnmfe algorithm, 1 file per day)

> analysisFiles (output of analysis files, usually saved as interim analysis data or to avoid re-running time-consuming analyses)

> PNR

> out (output of Save\_PNR to open and evaluate PNR manually by eye)

> tifFiles (tif files output of resave\_tiff\_fcn\_single used to create files in /out)

> tracker\_files (tracker .dat files, organized by day, created by position tracking Software)

> timestamps (timestamp .dat files, organized by day, created by miniscope DAcq Software)

> templates (template .mat files, created by alignment algorithm, 1 file per day)

FilenameMatch.xlsx

Create FilenameMatch.xlsx spreadsheet with the following info:

* day (recording day in real time)
* day count (recording day number)
* date
* animal
* session (e.g., HMC1, CYL1, RCT1, ...)
* videoname (filename, as created by the miniscope)
* trackername (e.g., m20\_cyl1\_day1\_20180213\_162340.dat)

This is the output of the extraction phase, the result of each is a set of matlab files for analysis as described in what follows:

cnmfe .mat output files (\*\_ss3):

contains file output.

output.A: spatial footprint of each cell

output.C: calcium trace of each cell (clean because output of the cnmf-e algorithm)

output.C\_raw: raw calcium trace of each cell used to infer C anc S

output.S: inferred spike trace of each cell

The following describes all the matlab and python functions used to analyze the calcium imaging and behavior output files.

SaveRegistrationByPairs.py

inputs:

FilenameMatch.xlsx

mat physiology file (to get spatial footprint of cells)

template files

Outputs to Data/CircleSquare/analysisFiles:

* AssignmentsByPairs\_M39.file

Output of this file is the registration of cells for recordings of separate days.

[assignments, [assignmentsByPairs, idPairs]]

cnmfe\_PlaceCells\_analysis.py

inputs:

FilenameMatch.xlsx

Through the loadPhysiology function:

* timestamps file (through read\_timestamps function)
* mat physiology file

Through Pos\_process function: Tracker files

AssignmentsByPairs\_M39.file

Outputs to Data/CircleSquare/analysisFiles:

* MapCorr\_12\_Bins\_M39.file

Output of this file is the categorization of each cell as PC or nPC along with the maps (2d and linear) and characteristics of each cell. Also contains tracking information and cell activity iused to compute the maps. Also contains the correlation of the maps across days.

[[maps\_corr, [maps\_corr\_pc\_any, maps\_corr\_pc\_all, maps\_corr\_pc\_not], day\_count\_pair, day\_real\_pair, sess\_pair, day\_sep, same\_diff, LinMapsCorr, LinMapsCorrPC, cellCount, trainingPair],

[day\_real\_list, day\_count\_list, sess\_list, rate\_maps\_list, PC\_idx\_list, LinMapsList, OccList, cohList, pInfoList, trainingList],

[X\_tracked\_list, Y\_tracked\_list, s\_tracked\_list],

[nRepeat]]

cnmfe\_temporal\_analysis.py

inputs:

FilenameMatch.xlsx

Through the loadPhysiology function:

* timestamps file (through read\_timestamps function)
* mat physiology file

Through Pos\_process function: Tracker files

AssignmentsByPairs\_M39.file

Outputs to Data/CircleSquare/analysisFiles:

* TempCorr\_List\_10000msAV\_1000msKT\_CircleSquare\_M39.file

Output is the temporal analysis of each day: activity vectors at specified time bins, tau correlations (with cell pair characeristics) computed using time series at bin size specified (time series also included)

[[day\_real\_list, day\_count\_list, sess\_list, S\_conv\_avc\_list, S\_conv\_tau\_list, isSeparatedList, tauPairsSingleList, tauVecSingleList, trainingList], [trackList, [], aMaxLocList], [atnSplitList, iDayList]]

* TempCorr\_Comb\_10000msAV\_1000msKT\_CircleSquare\_M39.file

[cv, tauVecAllComb, day\_count\_pair, day\_real\_pair, sess\_pair, day\_sep, same\_diff, isSeparatedAllComb, trainingPair, ratePairAllComb]

Activity vectors correlation matrices and pairs of tau correlation for matched recordings.

Cnmfe\_zscore\_analysis.py

Same as cnmfe\_temporal\_analysis.py with PTI instead of regular rate time series

Outputs to Data/CircleSquare/analysisFiles:

Zlinear\_List\_10000\_1000msKT\_CircleSquare\_M39.file

[[day\_real\_list, day\_count\_list, sess\_list, S\_conv\_avc\_list, [zConvTauList, S\_conv\_tau\_list], isSeparatedList, tauPairsSingleList, [zTauVecSingleList, tauVecSingleList], trainingList],

[trackList, [zBinnedRateMapsList, zRateMapsList], aMaxLocList], [atnSplitList, iDayList]]

Zlinear\_Comb\_10000msAV\_1000msKT\_CircleSquare\_M39.file

[cv, tauVecAllComb, day\_count\_pair, day\_real\_pair, sess\_pair, day\_sep, same\_diff, isSeparatedAllComb, trainingPair, ratePairAllComb]

Pos\_file\_processing.py

List of functions:

neighbor\_sum (rate\_map) 🡪 Spatial coherence

place\_info\_content (occ, rate\_map) 🡪 info content

nan\_gaussian\_filter (map, sig) 🡪 smoothed place map

read\_timestamps (filename) 🡪 read timestamps .dat files and outputs frame numbers and timestamps

Pos\_process (filename) 🡪 read dat files and outputs NumFrame, TimeFrame, Xpos, Ypos, Sector, Shock

conv\_nan (x,w): performs convolution ignoring NaNs

fisherZ (r) 🡪 z

loadPhysiology: function that inputs parameters and outputs phyiology mat and additional extracted and formatted information

SVM\_decode (s, sess, do\_cv, cv\_rep, cv\_params) 🡪 perf, coef, intercept, acc

runsTest (series, s0) 🡪 z-score

CS\_behavior

Plot dwell maps and traces for all recording for all animals and an animal average per recording. Also compute the distribution of the number of bins travelled by the animal in 1 s.

tauMapCompare.py:

Compare place map similarity values with tau cell-pair correlations, Also plot log-distribution of tau cell-pair correlations

tauZMapCompare.py:

Same as tauMapCompare for PTI and also compares tau with PTI

Isomap\_scratch.py

Isomap\_discardPercPop.py

Run Isomap transformation and remove or keep [option variable: discard] 5,10,25,50% cells with most cell pairs negatively or positively correlated [option variable: discardPop]

*discard*: True for removing % cells, False for keeping

*discardPop*: ‘pos’ for cells most positively correlated, ‘neg’ for megatively correlated, ‘kc’ for keeping all cells, ‘randPos’ for picking randomly

evaluatePTI.py

Compute difference between average activity of first half and second half of recroding (done for each cell, for each recording)

Below are functions we used to decode position using Support Vector Machine algorithm:

SvmDecodeTau\_loadTauSubSeg.py

SvmDecodeTau\_saveTauSubSeg.py

SvmLoad\_popSegments.py

SvmLoad\_tauSegments.py

SvmDecodeCs\_popSegments.py

SvmDecodeCs\_tauSegments.py

SvmDecodeCs\_match.py

Additional functions used for the isomap projections.

Isomap\_discardPercPop.py

Isomap\_scratch.py

Some additional plotting functions are below.

Count\_cells.py

modelForIntro

networkCoherence

plotFigure\_pc

plotFigure\_zPlaceMaps