**Olfactometry (pretty much everything I can tell you – T. Rust)**

Olfactometry/ - this folder is located on the synology server at (\\155.98.15.105\software)

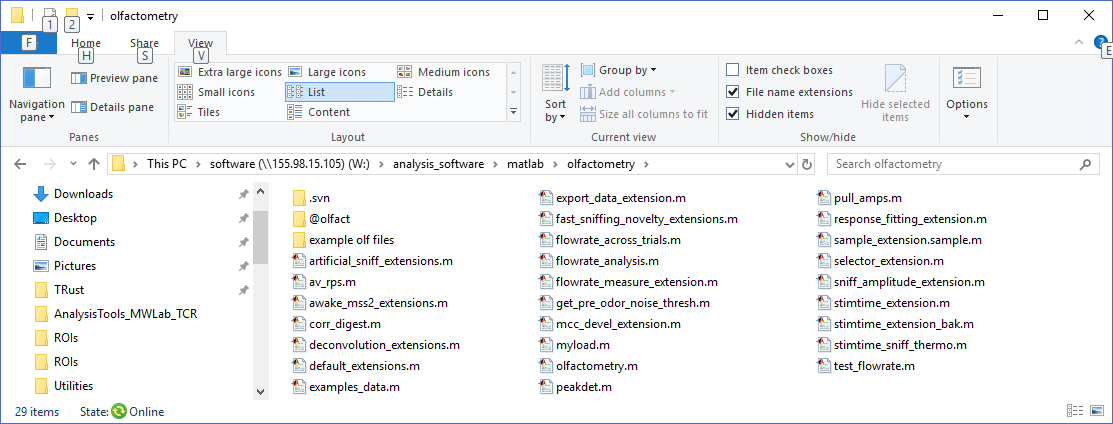
In folder “analysis\_software/matlab/olfactometry”. It also depends on the “/lib” and “/gui\_lib/”

folders located in the same directory. It seems to have been written using some Java Code interface methods adapted for MATLAB GUIs at the time it was written.

\*\*presumably, each odor presentation was acquired as separate images (trial) in previous work\*\*

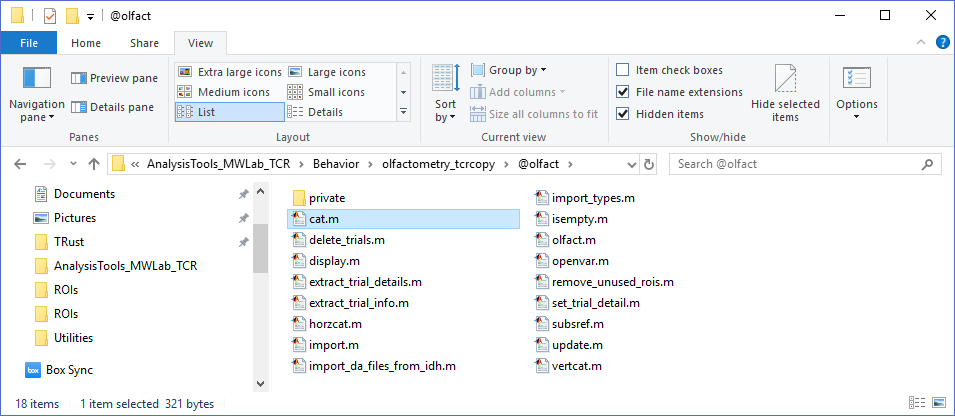
As you will see, it is a bit confusing… so we opted to basically start from scratch with a new behaviordataGUI and develop a data struct that works with our recent multi-odor/multi-trial data.

Here are the contents of the olfactometry folder:



*Olfactometry.m* is the main GUI program. The underlying data is structured according to the olfact class. The olfact class is defined in the class folder “@olfact”. A file with the extension “\*.olf” is a mat file with an olfact variable named “o” saved. To load these files to the MATLAB command window, use the command “load(‘filename.olf’,’o’,’-mat’).

Example: 1) run olfactometry; 2) using menu file>open> select rcr021.olf from folder named “example olf files\”; 3) select some rois and click on Views>default; 4) click Views>sniff/stim lines (sniff times will appear if present) 5) Click Analysis editing tools > Edit sniff/stim times to find the sniffs (you may need to do this before the last step if they were not already saved).



\*\*\*\*for more info about files in this directory see below \*\*\*\*

*olfact.m* is the class constructor file. The command “o=olfact;” returns an empty “Olfactometry object” which is a struct with fields: ‘trials’, and ‘rois’ (these are “properties” of the olfact class)

\* ‘trials’ is a struct with fields: ‘name’, ’timestamp’, ’numtrialavg’, ‘trial\_length’, ‘rois’, ‘other’, ‘measurement’, ‘measurement\_param’, ‘detail’, and ’import’

Olfact.trials.name –the name of the raw data file (e.g. neuroplex \*.da), -or- the “session name” followed by “time\_labels”

Olfact.trials.timestamp – timestamp (e.g. from header of .da file) or other

Olfact.trials.numtrialavg – presumably, some trials were averaged before import

Olfact.trials.trial\_length – duration of trial (secs)

Olfact.trials.rois – this is a struct that contains the traces (time series data) for each roi

Olfact.trials.rois.samplingrate

Olfact.trials.rois.datasize – frames x cols x rows, not sure about order

Olfact.trials.rois.nums – roi #s

Olfact.trials.rois.traces – \*\*matrix of traces, size (#rois x #frames)

Olfact.trials.rois.RLIs – \*\*matrix of resting light intensities (mean baseline fluorescence for each trace, time window hardcoded w/import seems to be frames 5:10, size (#rois x 1)

Olfact.trials.other – looks like the behavior data or ephys data

Olfact.trials.other.samplingrate – sampling rate of behavior data (1000 Hz)

Olfact.trials.other.odor\_onoff – odor signal

Olfact.trials.other.sniff\_thermo – sniff signal

Olfact.trials.other.lick – lick signal

Olfact.trials.other.odorvalence – odor valence signal

Olfact.trials.other.sniff\_pressure – (import\_ext\_file.m)

Olfact.trials.measurement – further defines behavior data

Olfact.trials.measurement.odor\_onset – time of odor on (based on threshold)

Olfact.trials.measurement.odor\_offset – time of odor off

Olfact.trials.measurement.odor\_onoff – Boolean true if signal is present

Olfact.trials.measurement.licktime – times of licks (based on some threshold)

Olfact.trials.measurement.licked – Boolean true if licked (e.g. between 3-8sec)

\*sometimes: measurement.odor\_onoff.odor\_onset/offset

Olfact.trials.detail – further defines behavior data

Olfact.trials.detail.comment – comments (e.g. import\_ext\_file.m)

Olfact.trials.detail.session –session name (e.g. import\_da\_files.m)

Olfact.trials.detail.odorant\_name

Olfact.trials.detail.odorant\_concentration

Olfact.trials.detail.odorant\_valence = ‘+’ or ‘-‘

Olfact.trials.measurement\_param

Olfact.trials.measurement\_param.odor\_onoff.thresh = 1000 (?mVolts)

Olfact.trials.measurement\_param.lick\_min = 3 (start of lick window)

Olfact.trials.measurement\_param.lick\_max = 8 (end of lick window)

\*sometimes: measurement\_param.licked.lick\_min/max

Olfact.trials.import – options used in the trial import

Olfact.trials.import.source.da\_dir – directory of .da files

Olfact.trials.import.source.det\_dir – directory of .det files

Olfact.trials.import.source.BNCs (from .da files)

\* ‘rois’ is a struct with fields: ‘name’, ‘source’, ‘index’, ‘points’, ‘position’, ‘measurement’, ‘measurement\_param’, and ‘detail’

Olfact.rois.index = roi#

Olfact.rois.points = indices where the roi mask == 1 (in roi) (a compact way to store rois)

Oflact.rois.position = [row,col] of roi centroid

\*\*\*\*more info about files in this directory \*\*\*\*

*cat.m* – just a copy of MATLAB’s cat function

*delete\_trials.m* – “method” function for deleting trials from olfact object

*display.m* – “overloaded” method to display the olfact struct (this just makes it difficult to see exactly what’s in the struct). You can type “tmp = struct(o)” to convert the olfact object to a struct and see what’s inside.

*extract\_trial\_details.m* – “method” to get the name, and some other info from trials.

*extract\_trial\_info.m* – “method” to get name, timestamp, trial\_length, numtrialavg from trials.

*horzcat.m* – concatenates olfact struct inputs

*import.m* – “method” to import data from da/det (neuroplex), ext (awake rat), and ofd (artificial sniff), and other files. Gets info from *import\_types.m* and calls “*import\_da\_files.m*” or “*import\_ext\_file.m*” or “*import\_behavior\_file.m*” or “*import\_ofd\_files.m*” (which are located in the “private/” subfolder.

*Import\_da\_files\_from\_idh.m* –function for adding some files to a olfact struct (not called by any other function as far as I can tell). This call “import\_da.m” from the “lib/” folder

*Isempty.m* - “method” to see if olfact object is empty

*Openvar.m* – “method” to automatically open an olfact object in the olfactometry gui

*Remove\_unused\_rois.m* – “method” to get rid of unused rois after trials are removed.

*Set\_trial\_detail.m* – “method” to add or change trial details subfield.

*Subsref.m* – “method” appears to access some index of olfact subfields

*Update.m* – “method” to update olfact fields and there values

Other files in main directory:

In the main /olfactometry directory, the files names “\*\_extension(s).m” do most of the “heavy lifting” for the olfactometry program. These are customized scripts which appear in the tree structure at the right hand side of the olfactometry GUI and do data processing on the olfact data. For example, the stimtime\_extension.m file can be used to find sniffs and generate the field olfact.trials.measurement.stim\_times, and the response\_fitting\_extension.m finds the responses in the traces for each roi and fits a double sigmoid to each sniff response and returns some things like sniff onset time and such. It’s all a bit convoluted to figure out since several types of analysis for several types of data seem to be all thrown together in this one program.

Here are some key pieces that seem relevant:

Default extensions (Views):

Default\_extensions.m is the place to find many things listed under Views in the tree structure on the right side of the olfactometry GUI. It’s a great place to start getting familiar with how this code is organized.

Sniff detection:

In the stimtime\_extension.m program – see the function “auto\_detect\_stim\_times\_Callback”

This takes 1 or more trials, figures out what type of data they are, and calls the function (at the end of the file) called “detect\_stims”. This has 3 different methods for finding sniffs depending on the type of data that’s present. For “sniff\_pressure”, you compute the z-score of the sniff trace, filter the signal, then look for places where that signal rises above a set threshold during a fixed time window.

Sniff responses:

In the response\_fitting\_extension.m function the detected\_stims are used to find the “response\_fits” – a window of time in the roi traces following each sniff is fit to a double-sigmoid function – which is then used to calculate the peak amplitude, onset time, and rise time between 10-90% of the fitted curve (note: any function will do if it is continuous, smooth, and matched the data). The double –sigmoid in this case has a weird factor of 4.4 thrown in to the time constant that just seems to help find a fit with fewer iterations. See the function dbl\_sigmoid\_arr near the end of the file, which looks like:

Function Y = dbl\_sigmoid\_arr(A,t)

Y= A(1)./exp(-4.4\*(t-A(2)./A(3)) +1) – A(4)./(exp(-4.4\*(t-A(5))./A(6)) + 1);

End

Note: the fitted parameters are not adjusted for the effect of the 4.4 (not sure why?).

Flow Rate:

The flowrate\_measure\_extension.m, and other flowrate extensions, appear to be designed to handle measurements of flowrate during inhalation derived from thermocouple measurements. I did not get into the details of these extensions. The data used are stored in olfact.trials.other.sniff\_thermo or sniff\_pressure, the method used is stored in trial.measurment\_param.stim\_times.method (appears to use the peakdet.m function), and results are saved in olfact.trials.measurement.flowrate with parameters in olfact.trials.measurement\_param.flowrate)

Other stuff:

As I understand it, the awake\_mss2\_extensions.m function was one of the most recently used (for the awake Rat papers, e.g. R.M. Carey et.al, 2009, *j neurophsiol*). This uses some of the stuff listed above, and has some additional calculations such as “the half-width of the deconvolved response to the first sniff after odor onset”, and “the Coherence stuff” which may not be directly applicable to the newer data with different fluorescence indicators, imaging rigs, and in mice – but it might be a place to find something useful.