Analysis (freely-walking)

Table of contents

1 Overview 1

2 Requirements for analysing the data from the MIC screen 2

2.0.1 Tree structure of processing functions . . . 2

2.1 Level 1 - analyse per cohort: `process_freely_walking_data` . . . 3

2.1.1 Inputs . . . 3

2.1.2 Outputs . . . 3

2.1.3 Description of `process_data_features` . . . 3

2.1.4 Explanation of the different functions used to combine data. . . . 5

2.2 Level 2 - analyse across cohorts: `process_screen_data` . . . 6

2.2.1 Inputs . . . 6

2.2.2 Outputs . . . 7

2.3 Level 3 - Statistical analysis of the data across cohorts: `make_summary_heat_maps_p27` 7

2.4 Processing of other protocols . . . 7

1 Overview

The data acquired from freely-walking optomotor experiments, especially during the screen using protocol 27, is analysed in two main steps.

The first step (`process_freely_walking_data`) is done per cohort (each vial of flies that was run). This creates several "overview" level plots for the individual cohort.

The second step (`process_screen_data`) combines data from across cohorts and parses the data based on the condition too. This creates plots that compare the behaviour of each strain against the empty-split control flies.

2 Requirements for analysing the data from the MIC screen

In order for the processing pipeline to run, within each experiment folder there should be:

- a .ufmf video of the entire experiment
- a .mat 'LOG' file
- a subdirectory that contains the file 'trx.mat' and the "-feat.mat" file outputted by FlyTracker.

The ufmf video is a compressed video format that is generated through the BIAS acquisition software. The difference between frames is stored, not the entire frame data. The LOG file contains metadata about the experiment (fly strain, date and time, which pattern was used for which condition) and it also contains the frame numbers at which each condition started and ended. This includes sub-parts of the conditions, such as when the stimulus changes direction. These frame numbers are recorded during the experiment by MATLAB interfacing with the image acquisition software, BIAS. The file 'trx' contains a MATLAB table array of the tracked data from FlyTracker. Each row in the table corresponds to an individual object that was tracked and the table contains columns of data about different behavioural metrics. All rows should have arrays of the same length that correspond to the total number of frames in the video.

~ What are the differences between feat and trx? When are they made during the tracking process? ~

2.0.1 Tree structure of processing functions

Functions in red are used for processing the data. Functions in blue are used for plotting the data.

- process_freely_walking_data
    - process_data_features
        - combine_data_one_cohort
        - make_overview
        - plot_all_features_filt
        - plot_all_features_acclim
        - comb_data_one_cohort_cond ***
        - plot_allcond_onecohort_tuning
        - plot_errorbar_tuning_curve_diff_contrasts
        - plot_errorbar_tuning_diff_speeds
        - generate_circ_stim_ufmf
            - create_stim_video_loop
- process_screen_data

  - comb_data_across_cohorts_cond
  - generate_exp_data_struct
  - plot_allcond_acrossgroups_tuning

2.1 Level 1 - analyse per cohort: `process_freely_walking_data`

2.1.1 Inputs

Requires string of the date for which you want to analyse the data. It will process all of the data from experiments conducted with any protocol that are within that day.

Runs the function `process_data_features` per cohort and experiment.

2.1.2 Outputs

- Exports a text file of the number of flies ran per protocol and per strain.
- Results .mat file
  - Contains: [LOG, feat, trx, comb_data, n_fly_data]
  - per vial
- Figures:
  - Acclim timeseries
  - Feat_overview timeseries
  - Timeseries per behavioural metric per vial.

2.1.3 Description of `process_data_features`

Processes the tracked data from FlyTracker.

Loads:

- LOG (metadata about the experiment.)
- feat (from FlyTracker)
- trx (from FlyTracker)

Saves in the results ".mat" file "_data.mat":

- LOG
- feat (updated with "poorly tracked" flies removed)
- trx (updated with "poorly tracked" flies removed)
- comb_data (combined data from all flies in the vial across the entire experiment)

- n_fly_data (3 x 1 array of [n_flies_in_arena, n_flies_tracked, n_flies_removed]). Useful to see how many flies were "lost" during the processing due to tracking errors.

1. **Combine the tracking data for all flies within one vial across the entire experiment**

   The data is not parsed based on condition yet.

   The function `comb_data_one_cohort` combines the data from all flies within a single experiment (vial of flies) into a single struct called `comb_data`. This struct contains fields for each behavioural metric (e.g. 'fv_data') and each field contains a 2D array of size [n_flies x n_frames]. This function takes in the output from FlyTracker (the 'feat' and 'trx' variables).

   The data is first checked for incorect or incomplete tracking. It runs the function `check_tracking_FlyTrk` which checks the rows of the table `trx` and removes any rows that do not contain the mode number of datapoints (frames).

   The data extracted directly from the **FlyTracker** output are:

   - distance from the edge of the arena (from `feat`).
   - heading (from `trx`).
   - x position (from `trx`).
   - y position (from `trx`).

   Data calculated from this data:

   - angular velocity (using the function `vel_estimate`)
   - forward velocity (two point, in direction of heading)
   - three point velocity in any direction (using the function `calculate_three_point_velocity`)
   - turning rate (angular velocity / forward velocity)
   - viewing distance (using the function `calculate_viewing_distance`)
   - inter fly distance (using the function `calculate_distance_to_nearest_fly`)
   - inter fly angle (using the function `calculate_distance_to_nearest_fly`)

   [PNG - example of this "comb_data" structure.]

2. **Create plots that give an overview of the behaviour of the flies during the entire experiment.**

   - Runs the function `make_overview` which generates a figure containing histogram subplots of the general behaviour of the flies over the entire length of the protocol.

   [PNG - example of this figure]

- Runs the function `plot_all_features_filt` which generates a plot of timeseries data for all flies over the entire length of the protocol. This figure is comprised of 4 subplots which show the forward velocity, angular velocity, turning rate and distance from the centre of the arena across the entire experiment. Coloured rectangles are used in the background to indicate when each condition occurred.

[PNG - example of this figure]

- Runs the function `plot_all_features_acclim` which generates a plot of timeseries data for all flies during the 5 minutes of acclimatisation in the dark. It uses the frame number that corresponds to the end of the acclimatisation period from the LOG file to determine which range of data to plot. The forward velocity, angular velocity, turning rate and both absolute and relative distance from the centre of the arena are plotted.

```
acclim_end = LOG.acclim_off1.stop_f;
range_of_data_to_plot = 1:acclim_end;
```

[PNG - example of this figure]

3. **Parse the behavioural data based on the conditions within the experiment.**

   The data from all flies is combined into an easier to manipulate 'struct' called `DATA` through the function `comb_data_one_cohort_cond`.

   [PNG - example of DATA structure.]

4. **Plot the data that has been parsed based on condition.**

   - Runs the function `plot_allcond_onecohort_tuning` which generates a [(n_conditions/2) x 2] subplot figure of the timeseries data during each condition (mean+SEM of all the flies in the vial) as well as tuning curves per condition.

   [PNG - example of this figure]

2.1.4 Explanation of the different functions used to combine data.

2.1.4.1 combine_data_one_cohort

This function is used within `process_data_features` to combine the data from all flies within a single experiment (vial of flies) into a single struct called `comb_data`. This struct contains fields for each behavioural metric (e.g. 'fv_data') and each field contains a 2D array of size [n_flies x n_frames]. This function takes in the output from FlyTracker (the 'feat' and 'trx' variables).

Critically, this function checks for flies with bad tracking and removes them. The tracked data returned from FlyTracker is also smoothed and checked for moments when the tracking might have gone wrong (when vel > 50 mms-1), those points are set to NaN and filled using an appropriate method for the datatype. This processed data is what is then saved to the results file and used to create the cross cohort 'DATA' struct used for all downstream analyses and plotting. The original data is never altered however and can be found in the original data folder.

2.1.4.2 comb_data_one_cohort_cond

Both `comb_data_one_cohort_cond` and `comb_data_across_cohorts_cond` create the nested data structure 'DATA' based on the conditions within the experiment. However, the single cohort version is **only** used within `process_data_features` to create the 'DATA' struct that is used for creating the overview time series plots that are saved per vial.

2.1.4.3 comb_data_across_cohorts_cond

This function is used within `process_screen_data` to combine the data from all flies across multiple cohorts (vials of flies) into a single struct called `DATA`. This struct contains fields for each behavioural metric (e.g. 'fv_data') and each field contains a 3D array of size [n_flies x n_frames x n_cohorts]. This function takes in the output from multiple runs of `process_data_features` (the 'comb_data' variable saved in the results .mat file). In order for this function to work, the experiments must have been run using a protocol that saves the condition number to the LOG file. This was not done for earlier protocols and so might cause trouble if trying to analyse earlier experiments.

2.2 Level 2 - analyse across cohorts: `process_screen_data`

This function works by using the .mat results files that were generated after running `process_freely_walking_data`. It heavily relies upon the structured format of the data into the same 'DATA' struct created by `comb_data_one_cohort_cond` but instead uses `comb_data_across_cohorts_cond` to generate the structure across all flies from multiple cohorts.

- Runs the function `comb_data_across_cohorts_cond` to generate the struct `DATA`.
- Runs the funciton `plot_allcond_acrossgroups_tuning` to create [(n_conditions/2) x 2] subplot figures for each strain versus the empty split control flies. It creates 5 of these subplot figures per strain, one for each data type ['fv_data', 'av_data', 'curv_data', 'dist_data', 'dist_data_delta'].

2.2.1 Inputs

- String of the protocol e.g. 'protocol_27'
- '.mat' results files from `process_data_features`.

2.2.2 Outputs

- 5 x figures per strain (timeseries per condition)
- Text file and 2 plots of the number of vials per strain and the number of flies per strain.

[PNG - example of the timeseries plots for one strain against the empty split flies]

2.3 Level 3 - Statistical analysis of the data across cohorts: `make_summary_heat_maps_p27`

This function generates a red-blue heatmap of the p-value for different behavioural metrics across each condition compared to the empty-split control flies.

- It combines all of the data for `protocol_27` in the same way as before using the function `comb_data_across_cohorts_cond`.
- It then uses the function `make_pvalue_heatmap_across_strains` to generate arrays of all of the p-values.
- A False-Detection Rate adjustment is performed using `fdr_bh`.
- The data is plotted altogether using the function `plot_pval_heatmap_strains`.

[PNG - example heatmap figure]

2.4 Processing of other protocols

Data from `protocol_30` and `protocol_31` run the functions `plot_errorbar_tuning_curve_diff_contrasts` and `plot_errorbar_tuning_diff_speeds`, respectively. Which plot tuning curve plots in addition to the timeseries plots.

Data from `protocol_25` - individual flies used the script `single_lady_analysis.m`.