Behavior Analysis Pipeline v2

**Raghuveer Parthasarathy**

Last modified: June 5, 2025

# Contents

History

Python code locations

Required Python packages

Required MAT and Excel files

# History

Version 1; based on Estelle Trieu’s code and my (major) modifications. Notes May 21-23, 2023

**Version 2:** Major revisions of everything – Re-writing all the code! Correcting errors; 7x speedup; etc. Begun May 2023. See “Behavior Code Revisions May 2023” … “… November 2023” documents.

# Python code locations

In progress versions in \Zebrafish behavior\Code\_current

Stable versions in \GitHub\Social-Behavior\BehaviorAnalysis

Documentation (incomplete) in \GitHub\Social-Behavior

# Required Python packages

numpy matplotlib pickle yaml csv pandas os xlsxwriter scipy time

# Required MAT and Excel files

These are the outputs from ZebraZoom output and some of the experiment parameters

**MAT files.** There must be one or more .MAT files containing zebrafish trajectory outputs, from ZebraZoom. Each MAT file may contain multiple behavior datasets. The .MAT files are typically named something like “results\_SocPrefInf\_2b\_t1.MAT”.

**Excel file.** The Excel file should contain headers like “Trial Trial\_ID, Pair\_ID, BodySize, Cond\_Code, PixelSize, Include”. The “Trial\_ID” and “Pair\_ID” are important for identifying the appropriate trajectory data in the .MAT file. “PixelSize” is the image scale, μm/px. “Cond\_Code” or something similar is an integer code indicating the experimental condition. The Excel file is typically named something like “SocPrefInf\_2b\_RaghuAnalysis.xlsx”.

**Location.** Place the above files in a folder, probably a sub-folder of “…\Zebrafish behavior\MAT files\”

# Processing tracking output (MAT → CSV)

## Generate CSV files

ZebraZoom’s output of fish body positions and headings in each frame is saved in MAT files as structured MATLAB arrays. We must convert this data to CSV files, one per dataset, for further analysis. (We will not use the ZebraZoom heading angles; see Sept. 2024 notes.)

The MATLAB program **convert\_fishTrackingData\_MATtoCSV.m** reads MAT files and writes data as a CSV file of the fish body positions, one per movie. There can be multiple MAT files, and each MAT file can contain multiple movies. Running this takes about 1 second per movie dataset. The output CSV files are in the same folder as the MAT file – see the next section about moving these files.

There are various inputs, shown below; the “Nfish” input is required, and is the number of fish in each movie of the behavior assay (in all data so far, this is either 1 or 2). The program checks that the number of fish in the datafiles matches this.

The program also reads the image offset positions, which are in the videoDataResults.wellPositions{k} structure for movie k as fields (topLeftX, topLeftY, topLeftX, topLeftY), and writes these to a separate CSV file (given by the wellOffsetPositionsCSVfile variable, probably “wellOffsetPositionsCSVfile.csv”), used in later analysis to determine the arena centers (see below).

**Example (MATLAB code):**

Nfish = 2;

dataDir = 'C:\Users\Raghu\Documents\Experiments and Projects\Misc\Zebrafish behavior\MAT files\2 week old - infected versus non-infected, solitary versus co-housed pairs';

wellOffsetPositionsCSVfile = 'wellOffsetPositionsCSVfile.csv';

cd 'C:\Users\Raghu\Documents\Experiments and Projects\Zebrafish behavior\Code\_current'

convert\_fishTrackingData\_MATtoCSV(Nfish, dataDir, [], wellOffsetPositionsCSVfile, []);

## Move CSV files

When finished, CSV files will be in the dataDir directory. We must move them to a new location, either a single folder or subfolders based on group or condition code.

This can be done manually or with code.

**Manual:** Move the CSV files to a folder (or group-based subfolders) in “\CSV files and outputs” . Copy the wellOffsetPositionsCSVfile.csv into each subfolder, and save a copy of the Excel log file of experiment information as a CSV file in each subfolder.

**Automated:** Run sort\_CSVfiles.py to copy and move files, and to sort by group or condition code, putting the appropriate CSV files into separate sub-folders. This also copies **wellOffsetPositionsCSVfile.csv**, copies the Excel log file of experiment information, and makes a CSV version of this Excel log file in each subfolder that contains only the row information for that subfolder. (It’s fine if **wellOffsetPositionsCSVfile.csv** contains information about datasets that aren’t present in that subfolder.)

Troubleshooting: If pair\_ID identification fails, ask for user input – probably the last number in the trial\_ID string. Also, if there aren’t subgroups, it may be easier to move/copy by hand.

CAUTION: Check manually that all CSV files were moved, and that “wellOffsetPositionsCSVfile” was copied. (Copy manually if not.)

# Required data and locations

Each dataset must have:

* **A CSV file of positions.** See “Generate CSV files,” above. The CSV filenames are used for the “dataset” variable names.
* **A CSV file containing experiment parameters, including the image scale.** Required for “Image Scale” and “Arena Offsets” below. Probably an Excel file originally; save as CSV. Automatically copied and split into subsets (groups) by process\_excel\_and\_csvs() – see above. The Excel and CSV files have a header row, but some earlier subsets might not – this should not cause problems. E.g. For 2023 datasets, **ArenaCenters\_SocPref\_3456.xslx** (**.csv**). For March 2024 Social / Co-housed, **SocDef\_Solitary\_AnalysisRaghu.xlsx (.csv).** Note that the CSV filenames of trajectories must match the names in the first column of this CSV, except for “\_light” and “\_dark” and other strings (like 'results\_', 'SocDef\_') specified in get\_ArenaCenter() in toolkit.py. For now, may need to adjust these strings manually.
* **An entry in the configuration file, all\_expt\_configs.yaml** . This must contain the frames per second (fps, almost certainly 25), arena radius (almost certainly 25 mm), the filename of the CSV file containing well offset positions (almost certainly wellOffsetPositionsCSVfile.csv), and more. See “experiment configuration file,” below.
* The configuration file also specifies the file in which the image scale is noted, imageScalePathName and imageScaleFilename, as well as the column index (0==first) that contains the image scale, imageScaleColumn.
* **Image scale.** The location of the image scale parameter is saved the location indicated in the experiment configuration file. For 2023 datasets, the image scale is noted in fifth column of **ArenaCenters\_SocPref\_3456.xslx** (**.csv**), mean ±- std = 56.7 ± 0.6 um/px . For 2024 cholic acid, for example, the scale is in “**SocPrefBA\_3b\_AnalysisRaghu.csv**” and is (mean ±- std = 96.4 ± 1.1 um/px). The column number is an image parameter that must be specified in the experiment configuration file (see below).
* **Arena offsets (well offsets).** These indicate the location of the arena (well) in the camera field. Saved as a row in a CSV file. Each row: CSV filename (without .csv) | topLeftX | topLeftY | lengthX | lengthY . The list “datasetRemoveStrings” in the experimental configuration file specifies strings to remove to find matches between the first column and the CSV file name (see below). The location of this information (filename, column) is noted in the experimental configuration file. (Likely default “wellOffsetPositionsCSVfile.csv”).
* **Arena centers.** These are the x and y positions of the centers of the arenas. The position values need the arena offsets to be subtracted to make sense. However, if arena centers aren’t noted they can be estimated from the well offset positions as topLeftX + lengthX/2 and topLeftY + lengthY/2 . For 2023 datasets, arena centers are in “ArenaCenters\_SocPref\_3456.csv” in the main folder, in columns 5 and 6 (0==first column). Note that this csv file has a header row. For most other datasets, the arena centers are estimated by the program. See arenaCentersLocation and arenaCentersColumns in the experimental configuration file; use arenaCentersLocation=None indicating that the center should be estimated.
* **Arena radius.** For 2-week-old fish, arena radius is always 25mm (50mm diameter). This parameter is saved in the experimental configuration file as arena\_radius\_mm .

# Folder structure

* Each experiment will have a “base” folder that either contains all the CSV trajectory files or that contains “subgroup” folders with the subgroup CSV files. It is recommended that the base folder is called “CSV files” and the program will warn the user if the name doesn’t contain “CSV files,” but this naming is not required.
* Each “base” folder must contain an experiment configuration file, called “expt\_config.yaml” . This configuration file describes this experiment only.
* Each “CSV files” folder must contain the configuration files that describe CSV structure and analysis parameters: analysis\_parameters.yaml, CSVcolumns.yaml

# Experiment Configuration File

Each experiment must have an experiment configuration file, written in YAML, in its “base” directory. This file must be called “expt\_config.yaml” and must have the following (note that some are optional, and the experiment name doesn’t matter):

**EXPERIMENT NAME:**

**expt\_name:** # experiment name

**fps:** # Frames per second (same for all in this experiment)

**arena\_radius\_mm:** # arena radius, mm (same for all in this experiment)

**imageScale:** # (optional) image scale, um/px, global (for all in this experiment) or leave empty to load from a file for each dataset

**imageScalePathName:** # Optional; leave empty if location is this folder (basePath). Path name of folder containing the CSV with image scale for each dataset; will append to basePath

**imageScaleFilename:** # Optional; file name of CSV with image scale for each dataset, if loading from file (i.e. if imageScale is empty)

**imageScaleColumn:** 4 # (optional) column (0-indexed) in CSV containing image scale for each dataset, if loading from file (i.e. if imageScale is None)

**datasetColumns:** [0,1] # columns to concatenate in the imageScale CSV to match dataset name (cat with "\_" between). For example, if [0, 1], will combine col 0 “SocDef\_Shank3\_1a\_t1” and col 1 “6” to be “SocDef\_Shank3\_1a\_t1\_6”

**datasetRemoveStrings:** # strings to remove from filenames and first column of the imageScale CSV to find matches. Also applies to the ArenaCenters CSV, if that's used. For example, file name may be “results\_SocDef\_Shank3\_1a\_t6\_6.csv”, and this will match dataset name “SocDef\_Shank3\_1a\_t1\_6” if datasetRemoveStrings is ['SocDef\_', 'results\_'] Removes these strings in the order of the list.

**offsetPositionsFilename:** # CSV file with well offset positions, almost certainly “wellOffsetPositionsCSVfile.csv”

# Arena center location information (optional; empty to estimate from well offset positions)

**arenaCentersPathName:** # # Path name of folder containing the arena centers file; will append to basePath. Leave empty to estimate from well offset positions

**arenaCentersFilename:** # CSV file name containing arena centers information

**arenaCentersColumns:** [5,6] # columns (0-indexed) with x,y arena centers

**subGroups:** # strings of subFolder names with data from experimental groups with the same overall parameters, e.g. ['Genotype 1', 'Genotype 2', 'Genotype 3']

# CSV column information file

There must be a configuration file that describes the columns of the CSV files, written in YAML. This file is “CSVcolumns.yaml” …\Zebrafish behavior\CSV files and outputs\ ; path and file name are hard-coded in behaviors\_main.py.

The contents are:

# Specify columns of the CSV files with fish trajectory information

**CSVcolumns:**

N\_columns: 26 # total number of columns in CSV file

head\_column\_x: 3 # head position x column (first col == 0)

head\_column\_y: 4 # head position y column (first col == 0)

angle\_data\_column: 5 # angle (radians) column; ignore (will re-calculate)

body\_column\_x\_start: 6 # starting column for body x positions (first==0)

body\_column\_y\_start: 16 # starting column for body x positions (first==0)

body\_Ncolumns: 10 # number of body datapoints

# Behavior Analysis Procedure

Analysis of a single experimental condition, consisting of multiple datasets.

Analyses all CSV files of body positions in a folder; one CSV file per dataset.

## Edit expt\_configs.yaml

If this experiment set hasn’t been examined before, name it and describe its image parameters in the configuration file expt\_configs.yaml.

See “Experiment Configuration File” above.

## Edit CSVcolumns.yaml

In the unlikely event that the structure of the CSV output has changed, edit the CSV column configuration file.

## Edit analysis\_parameters.yaml

Edit the configuration file with all the behavior analysis parameters (if you’re modifying these). Make sure the correct configuration file is written in the behaviors\_main.py file, at “# Get behavior analysis parameter info”.

Note that this includes the “base” names of the output files, probably “behavior\_counts.xlsx” etc., and the output subfolder, probably “Analysis”. Experiment and group names will be added to Excel output file names and the Analysis folder name.:

Excel file with summary count information: '[expt name]\_[SubGroup name]\_behavior\_counts.xlsx'

Analysis subfolder: 'Analysis\_[expt name]\_[SubGroup name]' instead of 'Group\_1'

The experiment folder name and subgroup name (if applicable) will be added to the output excel file names. For example, if allDatasets\_ExcelFile is behavior\_counts.xlsx, the experiment folder is “\2 week old - pairs assayed for gut transmission 0b” and the group is “Group\_2,” the behavior summary statistics file will be “behavior\_counts\_Group\_1\_2 week old - pairs assayed for gut transmission 0b\_Group\_1.xlsx”.

## Run behaviors\_main.py

This is the main program that loads CSV files, analyzes them, and outputs behavior characterizations. It takes about 4 seconds per CSV file to analyze.

**To use:**

* Run behaviors\_main.py
* When running for the first time on a set of experimental data, select loading option 1 to load from CSV files. When prompted, enter or select the “base” path name, i.e. the folder that contains CSV behavior files or that contains subgroup folders with the subgroup CSV files; see “Folder structure” above. The user can type the full folder path or leave it empty to select it using a dialog box. (Caution: the dialog box may be hidden.) The program checks that “CSV files” is in the lowest-level part of the path name (recommended but not required), and checks that “expt\_config.yaml” , “analysis\_parameters.yaml” and “CSVcolumns.yaml” are all in this “CSV files” folder (required).
* When prompted, enter the “experiment name” that will be appended to some output filenames. If an experiment name is in expt\_config.yaml, this is the default value.
* *Optional, strongly recommended:* Input the “base” name of two “.pickle” files to write, which will contain all the body position information (all\_position\_data) in one pickle file ({base pickle file name}\_positionData.pickle), the other of which will contain dataset information, analysis variables, and outputs ({base pickle file name}\_datasets.pickle). Default base pickle file namet: '[expt name]\_[SubGroup name].pickle'. If you don’t want to write pickle files, enter “none”.
* If running behaviors\_main.py again, position and other information can be loaded from the pickle files.

**Old:**

(global, all configurations.) Select the experiment. This will load parameters from the experiment configuration file. [finish writing this, make a global troubleshooting mode.]

## Outputs of behaviors\_main.py

The program will generate several files in the output\_subFolder directory (probably called “Analysis”):

* A summary text file for each CSV file with information including “basic properties” of the dataset (see below) and list of all frames with detected behaviors. Filename: same as the dataset name, .txt . Created by write\_behavior\_txt\_file(). Basic properties listed (all averaged over fish):
  + Experimental parameters (Image scale, frame rate, number of frames, …)
  + Tracking properties (Number of spans of continuous good tracking; mean length of spans, etc.)
  + Mean fish length
  + Mean fish speed
  + Mean fish speed when moving > threshold speed:
  + Bout rate
  + Mean bout duration
  + Mean inter-bout interval
  + Mean difference in fish length
  + Mean head-to-head distance
  + Mean closest distance
  + Fraction of time in proximity
* An Excel workbook called “behaviors\_in\_each\_frame + “\_” + [subGroup name] + “\_” [+ Experiment name] + “.xlsx” with a sheet for each dataset, indicating with an “X” behaviors (columns) found in each frame (rows). Created by
* A CSV file for each dataset that gives the basic measurements for each frame. Name: dataset + “\_basicMeasurements”.csv:
  + Head-to-head distance (mm) ["head\_head\_distance\_mm"]
  + The closest distance between fish (mm) ["closest\_distance\_mm"]
  + Speeds of each fish (mm/s) ["speed\_array\_mm\_s"].
  + Relative orientation angle of each fish with respect to head-to-head vector (radians) ["relative\_orientation"]
  + Distance to edge of each fish (mm)
  + Flag (0 or 1) for any fish being near the edge.
  + Flag (0 or 1) for any fish having bad tracking
* An Excel file with summary statistics of each behavior for each dataset, indicating in separate sheets the number of events, the duration (number of frames), the relative duration of each of the behaviors, and the durations relative to the number of frames in which any or all fish are “Active” (moving or bending). Each row is one dataset. Each column is one behavior; the first few columns are general dataset properties. In addition, separated by a blank row, each sheet contains statistics over the datasets (mean, std. dev., and s.e.m.). The Excel file is probably named “behavior\_counts” + “\_” + [subGroup name] + “\_” [+ Experiment name] + “.xlsx” . The base file name is specified in the analysis parameter configuration file, and the subgroup name if it exists is appended. This file is the most important output, and you may wish to rename it in the configuration file or afterwards.
* A .yaml file called all\_params.yaml with all the parameters used, merging expt\_config, analysis\_parameters, and the analysis/output path (dataPath).

**Pickle files**

If the user specified a base pickle file name (recommended), the program creates two pickle files:

* {base pickle file name}\_positionData.pickle , saved in the CSV folder (dataPath) containing all\_position\_data . This is a list of numpy arrays.
* base pickle file name}\_datasets.pickle , saved in the analysis outputs folder. Contain dataset information, analysis variables, and outputs. Includes datasets, CSVcolumns, expt\_config, params, and more.

When saving pickle files, the position data (\_positionData) pickle file is saved only if the data was read from CSV files; otherwise, it will not have changed. The program always saves the \_datasets pickle file.

# Loading datasets and variables from the pickle files

To load the contents of the pickle file, for example to make new plots based on datasets, use the function load\_and\_assign\_from\_pickle() in toolkit.py.

It requires that two pickle files have been saved (which in normal use of the program will have occurred):

(1) position data, probably in the CSV folder

(2) "datasets" and other information, probably in Analysis folder

The function will prompt for these file names. Press enter (no input) to get dialog boxes for selecting the .pickle files. Note that the dialog box may be hidden behind other windows! Recommended: close all figure windows.

leave empty (blank) for a dialog box. To avoid the dialog box, you can specify each pickle filename when prompted, either as just the filename or the full path + filename. If you just give the filename, you’ll be prompted for the folder.

From the Console:

from toolkit import load\_and\_assign\_from\_pickle

all\_position\_data, variable\_tuple = load\_and\_assign\_from\_pickle()

(datasets, CSVcolumns, expt\_config, params, N\_datasets, Nfish,

basePath, dataPath, subGroupName) = variable\_tuple

The last line is necessary to properly assign all the loaded variables. See behaviors\_main.py for an example.

# Output plots

## Single fish plots

make\_single\_fish\_plots(): Make plots of characteristics of individual fish, which may be in multi-fish experiments. Note that there are lots of parameter values that are hard-coded; this function is probably more useful to read than to run as-is; I suggest copy/pasting into the console and modifying its code.

**To use:**

Load the “datasets” information, as described above (“Loading datasets and variables from the pickle file”).

Load the function to make “single fish” plots:

from behavior\_identification\_single import make\_single\_fish\_plots

Run this, entering a “base” filename for saving figures, or None to skip; examples:

make\_single\_fish\_plots(datasets, outputFileNameBase = None)

or

make\_single\_fish\_plots(datasets, outputFileNameBase = 'all\_pairs\_light', outputFileNameExt = 'png')

**Plots**

* Speed histogram
* Radial position histogram – note that this is not normalized by r, so uniform distribution across a disk will give p(r) ~ r. This is intentional, to make the relative time spent at different r more apparent.
* Heading angle histogram (not interesting, but diagnostic)
* radial alignment angle – i.e. angle between heading angle and radial vector (not interesting, but diagnostic)
* Speed autocorrelation
* Bout Speed vs. time

## Pair assessment plots

Function make\_pair\_fish\_plots() in behavior\_identification.py . plots of characteristics of basic interactions of pairs of fish. Note that there are lots of parameter values that are hard-coded; this function is probably more useful to read than to run as-is; I suggest copy/pasting into the console and modifying its code.

**Plots:**

* Histogram of inter-fish distance (head-head distance)
* Histogram of inter-fish distance (closest distance)
* Relative heading angle histogram (i.e. the difference in heading angle between the two fish)
* Relative orientation angle histogram (i.e. angle between heading and head-to-head vector)
* 2D histogram of Speed vs. inter-fish distance. Note that this is dominated by low speeds, so manually adjust the color range. May need to try various ranges, depending on the dataset.
* 2D histogram of Relative heading angle vs. head-head distance
* 2D histogram of Relative orientation angle vs. head-head distance
* Speed of "other" fish vs. time during a bout
* Speed vs time during a bout, distance constraint of 5.0 mm. (Note that we could make this plot speed vs. time for the “other” fish by keyIdx = 'other' .)
* Speed cross-correlation. Note that we **can’t** impose a distance constraint.
* 2D histogram of C- and J-bend frequencies (combined) vs head-head distance. Note that the appearance of this will depend considerably on constraints and the color Range.

**To use:**

Load the “datasets” information, as described above (“Loading datasets and variables from the pickle file”).

Load the function to make “pair” plots:

from behavior\_identification import make\_pair\_fish\_plots

Run this, entering a “base” filename for saving figures, or None to skip; examples:

make\_pair\_fish\_plots(datasets, outputFileNameBase = None)

or

make\_pair\_fish\_plots(datasets, outputFileNameBase = 'all\_pairs\_light', outputFileNameExt = 'png')

# Comparison of Experiments

Comparison of datasets from two experimental conditions or groups, analyzing the single-experiment analysis outputs.

Python file: compare\_experiment\_behaviors.py

The Python file contains functions that compare the characteristics of behaviors, such as relative durations, for two different experiment groups, such as *2 week old light* and *2 week old dark*. The Excel file of summary statistics must exist for each experiment group. The code runs on each sheet in the Excel summary file, and outputs an Excel workbook with the same sheet names containing the comparisons. The code makes comparison plots and outputs an Excel file of statistical properties and comparisons. Allows excluding various behaviors (columns) from comparisons (hard-coded).

CSV output: property name, mean1, mean2, n1, n2, std1, std2, sem1, sem2, p\_mwu, p\_ks . The p-values are from a Mann-Whitney U test (tests the median being different) and Kolmogorov-Smirnov test (distributions being different).

Plots: Mean of each behavior for one experiment versus the other, as log-log plot (Set 2 on “y” axis, Set 1 on “x” axis); the ratio of the means of each behavior; and each behavior side-by-side for the two experiments.

**Outline of code:**

* Get Excel file names
* Determine what sheets the two Excel files have in common: get\_common\_sheets()
* Determine output file names
* Specify columns to exclude from analyses, plots
* For each sheet:
  + Read the sheet in each Excel file
  + Calculate statistics, ignoring any pre-existing statistics rows. Calculate statistics for each column, and ratio statistics (with bootstrap uncertainty).
  + Write results to Excel
  + Generate dataset 1 vs dataset 2 log-log plot
  + Generate ratio plot

**To use:**

Recommended: Close all figure windows. (They can cause problems with the dialog box.)

Run compare\_experiment\_behaviors.py. From the dialog boxes, select the first and then the second “behavior\_counts.xlsx” file – i.e. the Excel file in which relative durations of each behavior, along with statistics, are stored in a “Relative Durations” sheet.

When prompted either type the name of the output folder for plots and the CSV file, or leave this blank for a dialog box.

When prompted, enter name of the output Excel file.

When prompted, enter the “base” name of the output files for the plots, including the extension that specifies the image file type (e.g. eps or png). The program will add '\_relBehaviorPlot' and '\_relBehaviorRatios' to this, so for example entering “this\_expt.eps” gives outputs ““this\_expt\_relBehaviorPlot.eps” and “this\_expt\_relBehaviorRatios.eps”

Specify the various columns/keys (i.e. behaviors) to include in the comparison. Defaults are given, and a dialog box pops up to modify the selection. Note that you may need to expand the dialog box to see the “OK” button or the scroll bar. There are two lists, one for the “ratio” or “side-by-side” plots and one for the “log-log” plot.

Outputs:

* Excel file with statistics for each experiment and comparative statistics. This spreadsheet has the same sheet names as the input dataset spreadsheets, such as “Relative Durations.” Each column of the input datasets corresponds to a row in the Excel output file. The Excel output columns are:
* mean\_{1,2}, N\_{1,2}, std\_{1,2}, sem\_{1,2} for datasets {1,2},
* ratio of dataset 2/1 values and uncertainty and lower (16%) and upper (84%) percentile uncertainties, if these are calculated
* Statistical tests: MW and KS (see above)
* Several plots; see below.

Plots:

* Individual datapoints, “dataset 1 vs dataset 2”, along with mean and uncertainty.
* Ratio of dataset2 / dataset 1 for each behavior, arranged along the x-axis for each behavior, along with uncertainty.
* Individual datapoints for each behavior, arranged in pairs for dataset1 and dataset 2, along with mean and uncertainty.

# Combining experiments

We can combine information from pickle files of different experiments (combine\_expts\_from\_pickle()), or combine dataframes and Excel files (load\_and\_combine\_dataframes()).

**Combining datasets and other variables** as calculated in the program, merging into one composite experiment. Load the various pickle files, make sure analysis parameters are the same (e.g. frames per second, behavior parameters), and output one composite pair of pickle files (positions, datasets) for the concatenated variables. combine\_expts\_from\_pickle(). For details, see Code notes February 2025.

**To use:**

from toolkit import combine\_expts\_from\_pickle

combine\_expts\_from\_pickle()

Note that the first pickle file for each experiment is probably in the CSV folder and probably has a name ending in “\_positionData.pickle”. The second pickle file is probably in the Analysis folder, and and probably has a name ending in “\_datasets.pickle”.

The “experiment name” is almost certainly different for the different experiments; enter a new experiment name for the combined dataset.

Asks for the “base name” for the two pickle files to output. The program will add \_positionData.pickle and \_datasets.pickle to the base name

Also can write new Excel, CSV, YAML files based on the composite dataset – optional, asks user.

**Combine dataframes and Excel files:** load\_and\_combine\_dataframes(): Combine datasets’ outputs from two different experiments, concatenating information from each sheet (e.g. the relative durations) as recorded in the summary Excel file. Creates a new composite Excel file with the same sheet labels as the constituent Excel sheets. Optional outputs statistics, and optional output of statistical tests run to compare the two inputs, applied to each Sheet.

**To run:**

from compare\_experiment\_behaviors import load\_and\_combine\_dataframes

For example to output the summary statistics and not run comparison tests on the two datasets:

load\_and\_combine\_dataframes(outputStats = True, compareSets = False)

For example to output the summary statistics and run comparison tests on the two datasets:

load\_and\_combine\_dataframes(outputStats = True, compareSets = True)

# Modules and Functions

## toolkit.py

Functions for handling data files and performing various non-behavioral analyses

### get\_basePath()

Ask the user for the "base" Path that either contains all the CSV trajectory files or that contains "subgroup" folders with the subgroup CSV files.

### get\_loading\_option

Prompt the user to select a loading option (from CSVs or Pickle)

### get\_valid\_file()

Check if the file+path exists; if not, dialog box.

### load\_expt\_config()

Loads the experimental configuration file

Image scale and arena centers paths will be appended to config\_path (same as basePath in main code).

### load\_analysis\_parameters()

Loads the analysis parameters file

### check\_analysis\_parameters()

Checks that all the keys in the analysis parameters file exisand performs various checks

### set\_outputFile\_params()

Fill in keys in params corresponding to output folders, Excel file names

### get\_CSV\_filenames()

Select subgroup (if applicable) and get a list of all CSV files whose names start with startString, probably "results," in the basePath previously specified

### get\_dataset\_name()

Extract the "dataset name" from the CSV filename. Delete "results\_SocPref\_", "\_ALL.csv"; keep everything else

### load\_all\_position\_data()

For all CSV files in the list, call load\_data() to load all position data, and determine general parameters such as fps and scale

### load\_data()

Loads position data from a CSV file and returns a single array containing position information for all fish (position, angle, body markers etc.)

Works for any number of fish -- infers this from the first column

Also returns frame numbers (first column of CSV), checking that the frame number array is the same for each fish id of the dataset.

Checks that frame numbers are consecutive integers from 1 to Nframes for each ID; raises an Error otherwise.

### fix\_heading\_angles()

Fix the heading angles -- rather than the strangely quantized angles from ZebraZoom, calculate the angle from arctan(y[1]-y[2], x[1]-x[2]) . See notes Sept. 2024

### make\_frames\_dictionary()

Make a dictionary of raw (original) frames, frames with "bad" frames removed, combined (adjacent) frames + durations, total durations, and relative durations

Calls remove\_frames(),combine\_events()

### remove\_frames()

Remove from frames values that appear in frames\_to\_remove, and optionally dilate the set of frames to remove.

### dilate\_frames()

"dilate" the array of frame numbers.

### combine\_events()

Given an array of frame numbers, return an arrays of frame numbers with adjacent frames combined and duration numbers corresponding to the duration of adjacent frames.

### get\_output\_pickleFileName()

Get / construct pickle file name

### write\_pickle\_file()

Write Pickle file containing a dictionary of variables in the analysis folder

### load\_and\_assign\_from\_pickle()

Calls load\_dict\_from\_pickle() and assign\_variables\_from\_dict() to load \*two\* pickle files and assign variables.

### load\_dict\_from\_pickle ()

Load contents from Pickle file into a dictionary.

### assign\_variables\_from\_dict()

Assign dictionary elements loaded from pickle file, from load\_dict\_from\_pickle() to variables. Hard-coded variables; assumes pickle file contains datasets, CSVcolumns, expt\_config, params; returns these things,

### get\_Nfish()

Check that the number of fish is the same for all datasets; note this

### get\_edgeRejection\_frames()

Identify frames to reject in which the head position of one or more fish is close to the dish edge (within threshold). If there is no edge-rejection threshold, return empty numpy array

### get\_edgeRejection\_frames\_dictionary()

Calls get\_edgeRejection\_frames() to make a dictionary with frames for rejecting behavior, in which the head position of one or more fish is close to the dish edge (within threshold)

[HERE; FINISH. Line 651]

### load\_global\_expt\_config(config\_path, config\_file):

Loads the global experimental configuration file, which points to the experiment-specific configuration files

PLACEHOLDER – function to be written

## behavior\_identification.py

Zebrafish pair behavior identification functions

### def get\_basic\_two\_fish\_characterizations()

For each dataset, perform “basic” two-fish characterizations (e.g. inter-fish distance, relative orientation, relative heading alignment)

### get\_interfish\_distance()

Get the inter-fish distance, calculated both as the distance between head positions and as the closest distance, in each frame

### get\_close\_pair\_frames()

Find frames in which the fish are close to each other (closest distance < proximity\_threshold\_mm)

### extract\_pair\_behaviors()

Calls functions to identify frames corresponding to each two-fish behavioral motif in a single dataset.

### get\_contact\_frames()

Returns a dictionary of window frames for contact between two fish, which can be close distance between any fish body positions or head-body contact

# behaviors\_main.py

Contains the main analysis routine, which calls many other functions. Run this program for single-experiment analysis.

## main()

*Main function for calling data reading functions, basic analysis functions, and behavior analysis functions for all CSV files in a set.*

*Will read all CSV files or a previously written pickle file.*

*Writes a pickle file containing all trajectory and analysis outputs (“datasets” variable), and other variables – optional but strongly recommended.*

*Returns* datasets: *dictionary of all trajectory information and analysis*

**Outline:**

Two options for loading trajectories (prompt user):

1. Load position information from CSV files; do this if the dataset has not been previously analyzed.
2. Load position information from a pickle file. Also loads prior analyses, though these will be re-calculated.

If loading from CSVs:

* Load experiment configuration file; call load\_expt\_config(); dialog box if file not found.
* Get CSV column info from configuration file; dialog box if file not found.
* Get names of all CSV datasets: all CSV data files with names that begin with “Results”. Stores the file names in a list: “allCSVfileNames” . Calls get\_CSV\_folder\_and\_filenames()
* Get behavior analysis parameter info from configuration file
* Define output folders, modifying Excel output filename if there are subgroups
* Get pickle output filename
* Note number of datasets
* Set “showAllPositions” for displaying positions (diagnostic; typically False). If True, plotAllPositions() will be called to show all head positions, dish edge in a separate figure for each dataset.
* Call load\_all\_position\_data(). For all CSV files in the list, this calls load\_data() to load all position data and determine general parameters such as fps and scale.
* Note the number of fish (Nfish)
* Call repair\_head\_positions() to recalculate head positions based on indexes 1-3 (i.e. the 2nd, third, and fourth positions), since ZebraZoom’s head positions are unreliable.
* Call repair\_heading\_angles() to recalculate heading angles based on 2nd and 3rd body positions, since ZebraZoom’s heading angles are strangely quantized.
* Calculate fish lengths using get\_fish\_lengths()
* Call repair\_double\_length\_fish() to identify frames in which only one fish is tracked, see if this “fish” has twice the length a median fish (within some tolerance), and if so, split it into two fish. (Verify that the dataset’s Nfish==2 before calling.)

If loading a prior set of trajectories and analysis from a pickle file:

* When prompted, press enter (no input) to get a dialog box for selecting the .pickle file. Note that the dialog box may be hidden behind other windows! Recommended: Close all figure windows.
* Allow revision of the “experiment name,” which is used for various output names.
* Allow revision of the “output subfolder” so that output need not overwrite prior analysis

Next:

* Time reversal. Optional: avoid unless you’re sure of what you’re doing! Time-reverse one fish’s trajectory, same index (fish number) for each dataset.

For each dataset:

* Identify frames in which fish are close to the edge or in which tracking is bad; calls get\_edge\_frames(), get\_bad\_headTrack\_frames(), get\_bad\_bodyTrack\_frames()
* Also call make\_frames\_dictionary() to store this frame information.
* Call relink\_fish\_ids\_all\_datasets() to relink fish IDs across frames based on minimizing the Euclidean distance between body positions in consecutive good tracking frames.

For each dataset, identify characteristics or behaviors that involve single fish. Wrapper function get\_single\_fish\_characterizations(). This calls:

* Speed from get\_fish\_speeds(); also “isMoving” characterization, for each fish, and for “any” and “all” fish.
* Average speeds, for all good tracking frames and for the subset that “isMoving”
* Tail angle from getTailAngle()
* C-bends, J-bends from get\_Cbend\_frames(), get\_Jbend\_frames(). For C- and J-bends, exclude frames in which fish are close to the edge or in which tracking is bad; create a frames dictionary.

For each dataset, perform “basic” two-fish characterizations such as inter-fish distance and , relative orientation, if Nfish > 1. Wrapper function get\_basic\_two\_fish\_characterizations(). This calls:

get\_interfish\_distance() – Get the inter-fish distance – both the distance between head positions and the closest distance between any points – in each frame

get\_close\_pair\_frames()

calc\_head\_head\_vector() – calculate the head-to-head vector (for Nfish==2), in pixels, which will be used for relative orientation and other things.

get\_relative\_orientation()

get\_relative\_heading\_angle()– Get the difference on heading angle between the two fish, range [0, pi]. (Valid only for two fish.)

calcOrientationXCorr() – Get the sliding window cross-correlation of heading angles for the two fish

calc\_head\_head\_vector() – calculate the head-to-head vector (for Nfish==2), in pixels, which will be used for relative orientation and other things.

Also calculate the sum of the relative orientations, as an indicator of following behavior.

Also: the mean and std. absolute difference in fish length, the mean inter-fish distance, the mean angle-heading cross-correlation,

For each dataset calls extract\_behaviors() to identify and tabulate events; for each behavior, a dictionary containing frames.

Write count information to individual text files, individual Excel sheets, and summary CSV file. The CSV file that combines information from all datasets: allDatasetsCSVfileName . Writes each behavior counts and durations, and proximity-to-edge frames, and bad tracking (body) frames.

## extract\_behaviors()

Calls functions to identify frames corresponding to each behavior.

[write]

Identification of bad frames

# all\_position\_data: a list of numpy arrays

I save the array of body positions extracted from each CSV file in a list of numpy arrays.

* all\_position\_data[j]: a numpy array (Nframes x Ncolumns x Nfish) containing all the trajectory data for experiment j, from load\_data()

# datasets: a list of dictionaries

I save information other than the array of body positions in a list of dictionaries called “datasets”. There is one dictionary for each experiment. Dataset *j* for example will have its CSV filename (see Keys below) in datasets[j]["CSVfilename"].

Note that all\_position\_data[j] is the position information corresponding to datasets[j].

Some of these keys contain lists of dictionaries that I will call **behavior dictionaries**, which

**Keys for each experiment’s dataset:**

## Keys containing general experimental information or input data

* arena\_center: tuple of (x,y) positions of the Arena Center, from get\_ArenaCenter(), which reads center and offset information from CSV files.
* CSVfilename : the name of the CSV file containing all trajectory info.
* dataset\_name: the dataset name, extracted from the CSV filename by “get\_dataset\_name()”. (E.g. “3c\_2wpf\_k2” from “results\_SocPref\_3c\_2wpf\_k2\_ALL.csv” .)
* frameArray : 1D array containing all the frame numbers, from load\_data()
* image\_scale : the image scale (um/px), extracted from the appropriate CSV file by get\_imageScale()
* Nframes : the number of frames, set equal to the length of frameArray
* Nfish : the number of fish, set equal to datasets[j]["all\_data"].shape[2]
* total\_time\_seconds: total time (max frames – min frames + 1)/fps, seconds
* fps: frames per second, copied from expt\_config["fps"]

## Keys containing single-fish Information

* isActive\_Fish{j}: frames in which Fish *j* (*j* = 0, 1, 2, …, Nfish-1) has speed above threshold *or* bend angle above threshold, i.e. is “active.” Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames (dilated +1), not near-edge frame.
* isActive\_any : frames in which *any* fish has speed above threshold *or* bend angle above threshold, i.e. is “active.”. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames (dilated +1), not near-edge frames.
* isActive\_all : frames in which *all* fish have speed above threshold *or* bend angle above threshold, i.e. is “active.”. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames (dilated +1), not near-edge frames.
* bad\_headTrack\_frames : array of frames in which one or more fish head positions are bad (zero). From get\_bad\_headTrack\_frames().
* bad\_bodyTrack\_frames: array of frames in which one or more fish body positions are bad (zero). Get frame array from get\_bad\_bodyTrack\_frames(), then convert to a list of dictionaries (see **behavior dictionaries**, below) to also calculate runs, durations.
* bend\_angle: The angle of the second half of the fish body, relative to the first, defined such that a straight fish has bend angle 0. Radians. Array of shape Nframes x Nfish ; from calc\_bend\_angle().
* isBending\_Fish{j}: frames in which Fish *j* (*j* = 0, 1, 2, …, Nfish-1) has bending angle above threshold, i.e. is bending. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames, not near-edge frames .
* isBending\_any : frames in which *any* fish has bending angle above threshold, i.e. is bending. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames, not near-edge frames.
* isBending\_all : frames in which *all* fish has bending angle above threshold, i.e. is bending. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames, not near-edge frames.
* close\_to\_edge\_Fish{j}: frames in which Fish *j* (*j* = 0, 1, 2, …, Nfish-1) is within distance edge\_proximity\_threshold\_mm of the dish edge. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary excludes bad Tracking frames:
* close\_to\_edge\_any: frames in which *any* fish is within distance edge\_proximity\_threshold\_mm of the dish edge. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary excludes bad Tracking frames:
* close\_to\_edge\_all: frames in which *all* fish is within distance edge\_proximity\_threshold\_mm of the dish edge. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary excludes bad Tracking frames:
* bouts\_N: Number of bouts, from get\_bout\_statistics(), averaged over fish.
* bout\_duration\_s: Mean Duration of bouts, seconds, from get\_bout\_statistics(), averaged over fish.
* bout\_rate\_bpm: bout rate (bouts per minute), from get\_bout\_statistics(), averaged over fish
* bout\_ibi\_s: mean inter-bout interval (seconds), from get\_bout\_statistics(), averaged over fish
* Cbend\_Fish{j}, Cbend\_any: frames in which Fish *j* (*j* = 0, 1, 2, …, Nfish-1), or any fish shows a sharp bend (large bend angle). Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames, not near-edge frames.
* edge\_frames : frames in which any fish is close to the edge. Get frame array from get\_edge\_frames(), then convert to a list of dictionaries (see **behavior dictionaries**, below) to also calculate runs, durations.
* fish\_length\_array\_mm : length of each of the two fish in each frame (sum of all segments); mm; Nframes x Nfish array; from get\_fish\_lengths().
* fish\_length\_mm\_mean: mean fish length, mm, ignoring bad-tracking frames; a single value, averaged over both fish.
* fish\_length\_Delta\_mm\_mean and fish\_length\_Delta\_std: mean over frames of absolute difference in fish length, mm, ignoring bad-tracking frames, and standard deviation.
* heading\_angle: heading angle, re-calculated from 2nd and 3rd body positions. Nframes x Nfish array.
* Jbend\_Fish{j}, Jbend\_any: frames in which Fish *j* (*j* = 0, 1, 2, …, Nfish-1), or any fish shows a J-bend. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames, not near-edge frames.
* polar\_angle\_rad : polar angle of the head position relative to the arena center, for each fish in each frame; from get\_polar\_coords(). Note that “y” is defined as decreasing downward, so polar angle = atan(y,x) so polar angle = atan(y,x) increases Clockwise from East.. Range [-π, π]. Nframes x Nfish array. May include bad frames.
* radial\_position\_mm\_: radial position, i.e. distance from the head position to the arena center, for each fish in each frame; from get\_polar\_coords(). Nframes x Nfish array. May include bad frames.
* radial\_alignment\_rad : Angle between heading vector and radial vector, radians. If the fish is pointed away from the arena center, this is zero; towards the center, it is π (180 degrees). Note that this is simply the difference between the polar angle and the heading angle. Range [-π, π] Nframes x Nfish array. May include bad frames.
* Rbend\_Fish{j}, Rbend\_any: frames in which Fish *j* (*j* = 0, 1, 2, …, Nfish-1), or any fish shows a R-bend. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames, not near-edge frames.
* speed\_array\_mm\_s : speed of each fish, frame-to-frame, mm/second using scale and fps information; Nframes x Nfish ; from get\_fish\_speeds(). This is the frame-to-frame displacement, with speed[j] being position[j]-position[j-1], and the first element of speed set to zero so that there will be Nframes elements. (Note that zero for the first frame makes it easier to compare speed[j] and tail-angle[j] . May include bad frames.
* speed\_mm\_s\_mean: mean fish speed, mm/s, ignoring bad-tracking frames; averaged over both fish.
* speed\_whenMoving\_mm\_s\_mean: mean fish speed only for frames that meet the isMoving criterion, mm/s, ignoring bad-tracking frames; averaged over both fish.
* tail\_angle\_rad : tail angle of each fish, radians, calculated as the difference between the angle of the position 8-to-9 segment and the heading angle. (Use positions 8-9 rather than the final 9-10 because the latter segment is often very short.)
* isMoving\_Fish{j}: frames in which Fish *j* (*j* = 0, 1, 2, …, Nfish-1) has speed above threshold, i.e. is moving. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames (dilated +1), not near-edge frames.
* isMoving\_any : frames in which *any* fish has speed above threshold, i.e. is moving. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames (dilated +1), not near-edge frames.
* isMoving\_all : frames in which *all* fish have speed above threshold, i.e. are moving. Each is a list of dictionaries (see **behavior dictionaries**, below). This dictionary only excludes bad Tracking frames (dilated +1), not near-edge frames.

## Keys containing behavior information

* anyPairBehavior: Array of frames in which any identified pair interaction occurs: perpendicular orientation (all types), contact (any), tail rubbing, approaching (any), fleeing (any). See **behavior dictionaries**, below.
* approaching\_Fish{0,1}, \_any, \_all: Array of frames in which Fish 0 (resp. 1) is approaching Fish 1 (resp. 0); also arrays of frames in which any or all fish are approaching. From get\_approach\_flee\_frames(). See **behavior dictionaries**, below.
* approaching\_any and \_all :
* AngleXCorr\_mean, AngleXCorr\_std, AngleXCorr\_skew : mean, standard deviation, and skew of the xcorr\_array array. Only the mean is saved in the output CSV file.
* close\_pair: Array of frames in which the inter-fish distance is small (i.e. less than the threshold value of the "proximity\_threshold\_mm" parameter). A list of dictionaries (see **behavior dictionaries**, below. Note that this only excludes bad tracking frames, not near-edge frames.
* close\_pair\_fraction : Total fraction of time that the pairs are close (excluding bad tracking)
* closest\_distance\_mm: the inter-fish distance calculated as the closest distance between any inter-fish positions in each frame; mm; array of length Nframes. From get\_interfish\_distance() in toolkit.py.
* closest\_distance\_mm\_mean: mean closest inter-fish distance, ignoring bad-tracking frames.
* contact\_any: frames in which there is any contact behavior; a list of dictionaries (see **behavior dictionaries**, below).
* contact\_head\_body: frames in which there is head-body contact behavior, a subset of any contact; a list of dictionaries (see **behavior dictionaries**, below).
* contact\_larger\_fish\_head: frames in which there is head-body contact behavior and in which the head of the larger fish (only) is making contact with the other fish, a subset of head-body contact; a list of dictionaries (see **behavior dictionaries**, below).
* contact\_smaller\_fish\_head: frames in which there is head-body contact behavior and in which the head of the smaller fish (only) is making contact with the other fish, a subset of head-body contact; a list of dictionaries (see **behavior dictionaries**, below).
* contact\_inferred: frames corresponding to inferred contact, in which tracking is bad (zeros values) but inter-fish head-to-head distance was decreasing over some number of preceding frames and was below-threshold immediately before the bad tracking. The returned frame is the one immediately before the bad tracking. Make this a list of dictionaries, though this is technically unnecessary because by construction it can’t extend > 1 frame.
* fleeing\_Fish{0,1}, \_any, \_all: Array of frames in which Fish 0 (resp. 1) is fleeing from Fish 1 (resp. 0); also arrays of frames in which any or all fish are fleeing. From get\_approach\_flee\_frames(). See **behavior dictionaries**, below.
* head\_head\_distance\_mm: the inter-fish distance calculated as distance between head positions in each frame; mm; array of length Nframes. From get\_interfish\_distance() in toolkit.py.
* head\_head\_distance\_mm\_mean: mean inter-fish head-to-head distance (px), ignoring bad-tracking frames.
* 90deg\_noneSee: frames in which there is perpendicular orientation behavior in which the orientation is such that neither fish sees the other; a list of dictionaries (see **behavior dictionaries**, below).
* 90deg\_oneSees: frames in which there is perpendicular orientation behavior in which the orientation is such that one fish sees the other; a list of dictionaries (see **behavior dictionaries**, below).
* 90deg\_bothSee: frames in which there is perpendicular orientation behavior in which the orientation is such that both fish see the other; a list of dictionaries (see **behavior dictionaries**, below).
* 90deg\_larger\_fish\_sees: frames in which there is perpendicular orientation in which the larger fish (only) “sees” the smaller fish. A subset of 90deg\_oneSees. A list of dictionaries (see **behavior dictionaries**, below).
* 90deg\_smaller\_fish\_sees: frames in which there is perpendicular orientation in which the smaller fish (only) “sees” the larger fish. A subset of 90deg\_oneSees. A list of dictionaries (see **behavior dictionaries**, below).
* tail\_rubbing: frames in which there is tail-rubbing behavior, a subset of any contact; a list of dictionaries (see **behavior dictionaries**, below).
* xcorr\_array: Heading angle cross-correlation of the two fish over a sliding window for all frames. Value at a given frame is the normalized cross-correlation for the window frame ending at that frame. Nframes x 1 array.
* relative\_orientation: Angle between heading and head-to-head vector, for each fish. Also exported in the “basic measurements” file – see write\_basicMeasurements\_txt\_file(). May include bad tracking frames.
* relative\_heading\_angle: The difference in heading angle between the two fish (in range [0, π]), radians. (This is called “relative angle” in Stednitz 2024 If the fish are pointed in the same direction, this is zero; opposite directions, it is π (180 degrees). This is also exported in the “basic measurements” text file – see write\_basicMeasurements\_txt\_file(). May include bad tracking frames.

## Deleted

* circling : frames in which there is circling behavior; a list of dictionaries (see **behavior dictionaries**, below).

## Behavior dictionaries

For each behavior, make a sub-dictionary of datasets[j] that contains frames without “bad” frames, total durations, etc. The sub-keys of datasets[j][X] for behavior X (for example, X = "tail\_rubbing" ) are:

* behavior\_name: Name of the behavior (e.g. “tail\_rubbing”). The name of the behavior is the same as the key name, but this could be changed if desired.
* raw\_frames: 1D array of frame numbers in which this behavior was detected, or the first frame if it requires a frame window.
* edit\_frames: 1D array of frame numbers, the resulting of removing from “raw\_frames” any frames found in input lists of frames to remove using remove\_frames(), for example in which fish were close to the dish edge, or bad tracking frames.
* combine\_frames: 2D array in which the “edit\_frames” frames are replaced with the first frame of any run of sequential frame numbers (row 1 of the array) and the duration of the runs (row 2). Note that the length along axis==1 (columns) gives the number of runs, and therefore the number of events.
* N\_events: scalar. The number of events or runs (simply the length of the second row of combine\_frames).
* total\_duration: scalar, sum of the durations in row 2 of “combine\_frames”. This is therefore the number of total frames in which this behavior was detected, after deleting “bad” frames (dish edge, or bad tracking).
* relative\_duration: scalar, relative duration, i.e. total\_duration / Nframes. Again, deleting “bad” frames (dish edge, or bad tracking).

The rest of this module NOT YET MODIFIED BELOW HERE

Defines position, angle variables

Identifies circling events. Calls get\_circling\_wf → output circling\_wfs

Identifies 90-degree (perpendicular) events, including “one,” “both,” “none” subsets. Calls get\_90\_deg\_wf → output orientation\_dict and subsets “one,” “both,” “none” [should rename]

Identifies contact events, including “any” and “head-body” subsets. Calls get\_contact\_wf → output contact\_wf and subsets “any,” “head-body” [should rename “any”!]

Identifies tail-rubbing events. Calls get\_tail\_rubbing\_wf → output tail\_rubbing\_wf

Various outputs: text file, diagram, excel file:

Calls get\_txt\_file

Calls get\_diagram

Calls get\_excel\_file

**get\_90\_deg\_wf() in ninety\_deg.py**

Gets positions and angles for a given frame window

Calls get\_antiparallel\_angle to get angle (or cosine of angle) between fish headings; takes absolute value; compares this to the cos(theta) threshold.

Calls get\_fish\_vectors to get average heading angles over frame window.

Calls get\_connecting\_vector to get the normalized vector between fish head positions, connecting fish 1 and 2.

Calculate signs of cross product of fish vectors, and then call get\_orientation\_type in get\_90\_deg\_wf.py. to determine orientation type.

Evaluate criteria for 90 degree orientation: cos(theta) < threshold [rename variables, as noted elsewhere] and head-to-head distance < threshold. If so, append orientation type and frames.

**get\_tail\_rubbing\_wf() in tail\_rubbing.py**

Identifies tail-rubbing events.

Gets the 4 posterior-most body position markers for each fish. Also head positions, angles.

Calls get\_min\_tail\_distances “to get the minimum tail distances between two fish for two different body markers”

Evaluate the angle between fish headings, and the antiparallel criterion. Calls check\_antiparallel\_criterion → output [not saved]

If min tail distances and antiparallel criteria are met, append frames.

**Other functions in get\_90\_deg\_wf.py**

get\_fish\_vectors in get\_90\_deg\_wf.py. Returns a vector in the form (cos(theta), sin(theta)) for fish1 and fish2; average of heading angles over window size. [Simplify description to state this.]

get\_connecting\_vector in get\_90\_deg\_wf.py. Returns the normalized vector between fish head positions, connecting fish 1 and 2. [Check that it’s head positions.]

get\_orientation\_type in get\_90\_deg\_wf.py. Returns the orientation type of two fish given the sign of their respective (a, b, c) vectors.

# Write – behavior identification

## Contact functions

get\_contact\_frames()

get\_inferred\_contact\_frames()

## Perpendicular degree orientation functions

get\_90\_deg\_frames()

get\_orientation\_type()

## Tail-rubbing functions

get\_tail\_rubbing\_frames()

**Other functions in tail\_rubbing.py**

get\_min\_tail\_distances . Returns the minimum tail distances between two fish for two different body markers.

# toolkit.py

## Coordinate determination functions

get\_ArenaCenter(dataset\_name, arenaCentersFilename, offsetPositionsFilename):

Extract the x,y positions of the Arena centers from the arenaCentersFilename CSV -- previously tabulated. Image offsets also previously tabulated, first and second columns of offsetPositionsFilename

get\_edge\_frames(dataset, params, arena\_radius\_mm, xcol=3, ycol=4):

identify frames in which the head position of one or more fish is close to the dish edge (within threshold)

get\_imageScale(dataset\_name, imageScaleFilename):

Extract the image scale (um/px) from imageScaleFilename CSV -- previously tabulated

def estimate\_arena\_center(alldata, xcol=3, ycol=4):

Estimate the arena center position as the midpoint of the x-y range.

**Note:** This function is not currently used (July 4, 2023)

## Basic properties

get\_fish\_lengths(all\_data, CSVcolumns):

Get the length of each fish in each frame (sum of all segments)

## Bad Tracking identification and notation functions

def get\_bad\_headTrack\_frames(dataset, params, xcol=3, ycol=4, tol=0.001):

identify frames in which the head position of one or more fish is zero, indicating bad tracking

**Note:** This function is not currently used (July 4, 2023)

get\_bad\_bodyTrack\_frames()

identify frames in which tracking failed, as indicated by either of:

(i) any body position of one or more fish is zero, or

(ii) the distance between positions 1 and 2 (head-body) is more than

3 times the mean distance between positions j and j+1 for j = 2 to 9

remove\_frames(frames, frames\_to\_remove):

Remove from frames values that appear in frames\_to\_remove, and optionally dilate the set of frames to remove (e.g. for speed assessments).

## Visualization and diagnostic functions

# Visualizing fish trajectories and information

Interactive visualization of fish body positions, headings, and other information. Keyboard controls are displayed when starting; also:

Keyboard Controls:

u: +1 frame t: -1 frame

i: +10 frames r: -10 frames

o: +100 frames e: -100 frames

p: +1000 frames w: -1000 frames

v: toggle the text display of information: heading angle and “additional information” – see below

z: toggle autoscaling (on/off)

Example:

Dataset 3b\_k1, frame 686:

A graph with blue and gray lines

AI-generated content may be incorrect.

**Usage:**

datasets, all\_position\_data, and CSVcolumns should be loaded.

If CSVcolumns is not loaded, can load with:

basePath = r'C:\Users\raghu\Documents\Experiments and Projects\Zebrafish behavior\CSV files and outputs\2 week old - pairs TestSubset'

CSVinfo\_file = 'CSVcolumns.yaml'

CSVinfo\_file\_full = os.path.join(basePath, CSVinfo\_file)

with open(CSVinfo\_file\_full, 'r') as f:

all\_CSV = yaml.safe\_load(f)

CSVcolumns = all\_CSV['CSVcolumns']

To run the visualization:

from fish\_position\_visualizer import visualize\_fish\_data

j=0

vis = visualize\_fish\_data(all\_position\_data[j], datasets[j]["heading\_angle"], datasets[j]["head\_head\_vec\_px"], CSVcolumns["body\_column\_x\_start"], CSVcolumns["body\_column\_y\_start"], CSVcolumns["body\_Ncolumns"], additional\_info = datasets[j]["relative\_orientation"])

See “Code notes May 2025.docx” for more information about usage, loading variables.

**OLD; not useful:**

plotAllPositions(dataset, CSVcolumns, arena\_radius\_mm, arena\_edge\_mm = 0):

Plot head x and y positions for each fish, in all frames also dish center and edge

visualize\_fish(body\_x, body\_y, frameArray, startFrame, endFrame, dataset\_name):

Plot fish body positions (position 1 == head) over some range of frames body{x, y} are Nframes x 10 x Nfish=2 arrays of x and y positions

## Data export

Note that the summary CSV of all datasets is created in behaviors\_main.py

write\_output\_files(params, dataPath, datasets):

Calls other functions to write the output files (several) for all datasets

Hard-codes a superset of all keys (behaviors) to write, and then removes any keys that are not in the first dataset, for example two-fish behaviors if that dataset was for single fish data

write\_behavior\_txt\_file(dataset, key\_list):

Creates a txt file of the relevant window frames and event durations for a set of social behaviors in a given single dataset

Output text file name: dataset\_name + .txt

mark\_behavior\_frames\_Excel(markFrames\_workbook, dataset, key\_list):

Create and fill in sheet in Excel marking all frames with behaviors found in this dataset

## Correlation functions

calculate\_autocorr()

calculate\_block\_autocorr()

calculate\_crosscorr()

calculate\_block\_crosscorr()

calculate\_value\_corr\_oneSet()

calculate\_value\_corr\_all()

## Combining and plotting data

combine\_all\_values\_constrained(): Combine all values of one characteristic, like speed, across all datasets. Optional: can combine values of one characteristic subject to a mask on another characteristic, for example all speed values when inter-fish distance is close. Ignore, in each dataset, "bad tracking" frames. For example: get all speed values for frames in which inter-fish-distance is below 5 mm.

*August 27, 2024:* Allow for constraints properties to be arrays with more than one fish, e.g. for speed, either taking values for one of the fish, or an average, or max or min.

Can then use combined values in, for example, plot\_probability\_distr().

See below (plot\_probability\_distr) for examples.

## Probability distributions and histograms

plot\_probability\_distr(): Plot the probability distribution (normalized histogram) for each array in x\_list (semi-transparent) and for the concatenated array of all items in x\_list (black). Typically use this with the output of combine\_all\_values\_constrained() . For example:

r\_mm\_all = combine\_all\_values\_constrained(datasets, keyName='radial\_position\_mm', dilate\_plus1 = False)

plot\_probability\_distr(r\_mm\_all, bin\_width=1.0, bin\_range = [-0.5, 25.5], xlabelStr='Radial position, mm', yScaleType = 'linear', flatten\_dataset = False)

r\_mm\_all\_close = combine\_all\_values\_constrained(datasets, keyName='radial\_position\_mm', constraintKey='head\_head\_distance\_mm', constraintRange=[0.0, 5.0], dilate\_plus1 = False)

plot\_probability\_distr(r\_mm\_all\_close, bin\_width=1.0, bin\_range = [-0.5, 25.5], xlabelStr='Radial position, mm', yScaleType = 'linear', titleStr = 'Radial position, if close fish', flatten\_dataset = False)

Note that this **the probability distribution of for p(r) isn’t normalized by 1/r;** this is intentional, to make the relative time spent at different r more apparent.

The function can also plot the histogram in polar coordinates – useful for angle distributions.

make\_2D\_histogram(): Create a 2D histogram plot of the values from two keys in the given datasets. Can specify ranges, dilation of “bad frames” – always avoid bad frames. Uses combine\_all\_values\_constrained() to combine across all datasets the values of some characteristics, removing (dilated) bad frames, etc. See August 2024 notes, including August 27, 2024 notes – can use particular fish for values, and for constraints.

**Aligning and merging bout trajectories**

August 21-22, 2024; revise Oct. 20, 2024

average\_bout\_trajectory\_oneSet(): Tabulates speed information from dataset["speed\_array\_mm\_s"] around each onset of a bout ("isMoving" == True) in the time interval specified by t\_range\_s. Averages these for all bouts for a given fish; considers each fish in dataset. Optional: only consider bouts for which the value of constraintKey at the start frame (first isMoving frame) is between constraintRange[0] and constraintRange[1].

average\_bout\_trajectory\_allSets(): For all datasets, call average\_bout\_trajectory\_allSets() to tabulate speed information around each onset of a bout ("isMoving" == True) in the time interval specified by t\_range\_s. Averages these for all bouts, each fish and each dataset. Optional: only consider bouts for which the value of constraintKey at the start frame (first isMoving frame) is between constraintRange[0] and constraintRange[1]. See also Aug. 27, 2024 notes – can use particular fish for constraints.

plot\_function\_allSets()

behaviorFrameCount\_one()

behaviorFrameCount\_all()

# behavior\_identification\_single.py

## Behavior identification or characterization functions that apply to single fish

get\_fish\_lengths()

get\_fish\_speeds()

get\_mean\_speed()

getTailAngle()

getTailCurvature()

get\_bout\_statistics()

get\_Cbend\_frames()

get\_Jbend\_frames()

combine\_all\_values():  Loop through each dataset, get values of some numerical property, and collect all these in a list of numpy arrays. Ignore, in each dataset, "bad tracking" frames. If "dilate\_plus1" is True, dilate the bad frames +1; do this for speed values, since bad tracking affects adjacent frames! List contains one numpy array per dataset (possibly with multiple columns corresponding to fish). Output can be used, for example, for making a histogram of speeds or inter-fish distance.

plot\_probability\_distr(): Make a histogram of values from list of values, for example all speeds as generated by combine\_all\_values():.

get\_fish\_lengths\_mean\_gmm(): Fit each fish's length data, for good tracking frames only, to a Gaussian mixture model and return the mean of the component near the median value.

# load\_and\_visualize\_behaviors.py

Functions for loading data from a pickle file, and visualizing (diagnostics)

Loading data from a pickle file

loadAllFromPickle()

# Abandoned functions

Move to an “abandoned” folder

get\_reorientation\_wfs() in matching.py: reorientation “matching” behavior

get\_mirroring\_wfs(): in mirroring.py: "mirroring" behavior.

get\_circling\_frames(): “circling” behavior – lots of failed attempts