fishproviz: Fish Tracking Data Processing and Visualization

Project 21: Developing Exploration Behavior

Tracking genetically identical individuals from day 1 of their life to understand the development of intelligent behavior

Installation

Requirements

- Unix-like Operating System
- python3
- gcc
- latex

Installing dependencies

```
python3 -m pip install -r requirements.txt
```

Additional development dependencies

```
python3 -m pip install pre-commit
pre-commit install # installation of git hooks
```

Build

For setting up the project first and creating a local environment file `fishproviz/config.env`, run:

```
python3 setup.py build_ext --inplace
```

Getting Started

Configuration

For specifying hyperparameters for the usage of the program, it is neccessary to insert the necessary variables and paths in the following configuration file: fishproviz/config.env Entry-Points for the Dataset for running the program locally are given with the following env-variables: - `path_csv_local`: root folder of the data directory - `POSITION_STR_FRONT`: specified sub-directory for the front compartment - `POSITION_STR_BACK`: specified sub-directory for the back compartment

General Usage

```
usage: python3 main.py [-h] [-ti TIME_INTERVAL] [-fid FISH_ID] [--include_median]
                       {trajectory,feeding,trial_times,activity,turning_angle,
                       abs_angle,tortuosity,entropy,wall_distance,all,clear}

This program computes metrics and visualizations for fish trajectories,
the results are saved in the directory:
'DIR_CSV_LOCAL'

positional arguments:
  {trajectory,feeding,trial_times,activity,turning_angle,abs_angle,
  tortuosity,entropy,wall_distance,all,clear}
                        Select the program you want to execute

options:
  -h, --help            show this help message and exit
  -ti TIME_INTERVAL, --time_interval TIME_INTERVAL
                        Choose a time interval in second to compute
                        averages of metrics. Also possible [day, hour].
  -fid FISH_ID, --fish_id FISH_ID
                        Fish ID to run can be set by 'camera_position'
                        or index, default is all fish_ids
  --include_median      Include median or not only for activity
  -logs, --print_logs   Print logs

Example of use: python3 main.py trajectory -fid 0
```

1. Trajectory Visualization

Configuration Prerequisite: please make sure that the input *csv*-files are pre-filtered regarding possible nan values due to incorrect tracking/not available fishes

Necessary Variables: - `N_BATCHES`: number of used batches - `path_csv_local`: root folder of the data directory - `POSITION_STR_FRONT`: specified sub-directory for the front compartment - `POSITION_STR_BACK`: specified sub-directory for the back compartment

Plot Generation for Activity Analyses For generating single pdf-files with plots with trajectories for every timeframe run the following command, a parallel execution is recommended but not required, the argument `parallel` accepts the values "True" or "False":

```
python3 main.py trajectory --parallel "True"
```

The individual pdf-files are stored in `./visualisations/plots/trajectory`.
Trajectory Logs are stored in the directory `./logs` and include information about erroneous data that has been filtered out.

Bundling these locally created individual timeframes into one combined file is possible using the following bash-script:

```
bash scripts/build-trajectories.sh -l #-l flag for using the local configurations
```

The resulting bundeled pdf-files for every individual are then stored in `./visualisations/trajectory`

Usage on the Lab Computer - Trajectories As the lab computers use windows, an instance of the Windows Subsystem for Linux needs to be started on the Left PC. This is configured there with Ubuntu as the target Linux-Derivate. 1. open Ubuntu in WSL on Windows by typing into the terminal: `bash` 2. check, whether the external drive is correctly mounted into ubuntu and expect not an empty folder to appear:

```
ls /mnt/e
```

- if the folder is unexpectedly empty, the drive seems not to be mounted correctly. Mounting it requires administration-priviledges, for that, please contact your supervisor. Afterwards execute the following command:

  ```
  sudo mount -t drvfs e: /mnt/e
  ```

- the mounted drive is dependent on the variable assigned by windows, in this ongoing example this drive is referenced with `e`. please note, that creating the directory in `mnt` is neccessary, if this has not already been created. this can be done with the following command:

  ```
  sudo mkdir /mnt/e
  ```

3. change the active working directory to

```
cd ~/project21/fishproviz
```

4. open the file `fishproviz/config.env` to configure the environment file and set the path variables that point to your working directory

```
notepad.exe fishproviz/config.env &
```

- please note, that linux doesn't accept whitespaces or special characters (like braces) in file names. These need to be escaped with a backslash
- note that the prefix to the different data sources is `/mnt/e`.

5. for calculating trajectories, run

```
python3 main.py trajectory
```

6. bundling each trajectory file into one trajectory-file per individual using:

```
bash scripts/build-trajectories.sh -l #-l flag for using the local configurations
```

- if the execution of this bash-script results in permission errors, check the access-rights for this files using

```
ls -las scripts/build-trajectories.sh
# output: >>> 16 -rw-r--r--  1  user user-group date time scripts/build-trajectories.sh
chmod u+x scripts/build-trajectories.sh # users receive execution-rights for this file
ls -las scripts/build-trajectories.sh
# output: >>> 16 -rwxr--r--  1  user user-group date time scripts/build-trajectories.sh
```

Plot Generation for Feeding Trajectories For generating single pdf-files with plots with trajectories for every timeframe, run the following command: - Feeding Trajectories: `python3 main.py feeding` The CSV-file for time spend, feeding and number of visits are stored at `results/feeding`. - Optional requirement: Provide a csv-file (;-separated) with start and end time for the feeding measures with columns in the following format:

day	time_in_start	time_in_stop	time_out_start	time_out_stop
dd.mm.yy	hh:mm	hh:mm	hh:mm	hh:mm

```
# fishproviz/config.env
SERVER_FEEDING_TIMES_FILE="path/to/feeding_times.csv"
SERVER_FEEDING_TIMES_SEP=","
FEEDING_SHAPE="patch" # or "ellipse"
FEEDING_SHAPE_WIDTH=5
FEEDING_SHAPE_HEIGHT=5
MAGNET_LENGTH_CM=2.5
FEEDING_PATCH_COORDS_FILE="data/feeding_patch_coords.csv"
FEEDING_PATCH_COORDS_SEP=","
```

```
- Feeding Shape `ellipse`:
    -   Requirements: Set the `path_recordings` in `config.env` to the correct path where the recordings
        and annotations are stored containing the data of the feeding ellipses. This is usually on the
        server, make sure you connect to the server to access the data for the first time.
    After the first run, the data is stored locally in `config_data/feeding_zones` and the server is not
    needed anymore.
    -   TODO: In the future we want to store the feeding zones in the `path_csv_local` folder and not
        with the recordings.
- Feeding Shape `patch`:
    -  Requirements: Provide a csv-file path with the coordinates of the feeding patches through
       FEEDING_PATCH_COORDS_FILE. FEEDING_PATCH_COORDS_SEP indicates the delimitor. The file used is
       [data/feeding_patch_coords.csv](data/feeding_patch_coords.csv). If modified patch coordinates are
       needed change the path in the program `feeding_shape.py`. Patch width and height in centimeters
       should be specified in the configuration file through `FEEDING_SHAPE_WIDTH` and
       `FEEDING_SHAPE_HEIGHT`, respectively.
An example template can be found at

[data/recordings_feeding_times_template.csv](data/recordings_feeding_times_template.csv)
```

Bundling these locally created individual timeframes into one combined file is possible using the following bash-script:

- `bash scripts/build-trajectories.sh`
- Optional argument:
  - `--feeding` or `-f` for the feeding trajectories.
  - `--test`, `-t` is used to test the script, to generate only the fist pdf.
  - `--local`, `-l` to use the paths of the local hard drive to link the csv file in the pdf.
  - `--cam-id, -cam` followed by `cameraID_position`, to create only the pdf for the given camera.

Remark: For the bash-script you can not build feeding and non feeding trajectories in parallel as they use the same files.

Usage on the Lab Computer - Feeding

1. the configurations for feeding are analog to the trajectory configuration, please refer to the steps 1-4 in the trajectory section
2. for calculating trajectories, run

   ```
   python3 main.py feeding
   ```

3. bundling each trajectory file into one trajectory-file per individual using:

   ```
   bash scripts/build-trajectories.sh -f -l  #-l flag for using the local configurations
   ```

2. Data File and Path Validation

The python script `path_validation.py` is used to validate the filenames and paths of the data files. It logs all error messages into `log-path-validation.txt`.

```
usage: python3 path_validation.py [-h] [--delete] [--n_files N_FILES] [--path PATH]
options:
```

```
  -h, --help         show this help message and exit
  --delete           If set, the duplicates will be deleted.
  --n_files N_FILES  Number of files to expect in each folder, default is 15, for feeding use 8, for a log
                     file that is cleaner.
  --path PATH        Path to the directory that contains the folders front and back, default is
                     /Volumes/Extreme_SSD/test_tracks.
```

3. Trajectory Analysis

- run: `python3 main.py <<metric>>`
- For **metric** use one keyword out of:
  - `activity, turning_angle, tortuosity, entropy, abs_angle, wall_distance`.
- run `python3 main.py <<metric>> --time_interval <<hour/day>>` to record mean and standard derivation per fish per hour/day in one csv-file.

3.1 Metrics:

- step length is the length of the vector drawn between to consecutive data frames.
- the mean and standard derivation illustrated in the visualization is computed from filtered data frames, removing obvious error point and normed by the distance between data frame when erroneous data point where removed.
- The number of spikes is defined by the threshold of ‘ > 10‘cm in one step. The threshold can be changed in the config.env file.
- For the sum of angles we take each angle between consecutive steps anti-clockwise ‘$\alpha \in [-\pi, \pi]$‘.
- For the average angle each angle ‘$\alpha > 0$‘

4 Data Visualizations

4.1 Entropy Density

- run: `python3 -m fishproviz.visualizations.entropy_plots` *plotly needs to be installed*
- run: `bach scripts/entropy_density.sh` The PDFs with show in tex/entropy__density

4.2 Metrics over 4 Weeks

- run: `python3 main.py program=all time_interval="day"` to calculate all metrics by day an save them to a csv
- run: `python3 -m fishproviz.visualizations.activity_plotting` to plot the data of the csv-files.
- run: `bash scripts/metrics.sh` to create the summery PDF.

File Structure

The variable *path_csv_local* in fishproviz/config.env is the root of the project and the place where all generated data is stored. In addition to the initial front an back directory where all the tracking data is stored you will find the following directories after the corresponding program executes.

Folder Structure:

```
.
|-- path_csv_local          # root folder of the data directory
|   |-- front               # POSITION_STR_FRONT front compartments tracks
|   |-- back                # POSITION_STR_BACK back compartments tracks
|   |-- area_config         # area_config Area Configurations
|   |   |-- front           # front
|   |   |-- back            # back
|   |-- visualizations      #
|   |   |-- trajectory      # trajectories pdfs from latex
|   |   |-- feeding         # feeding trajectories pdfs from latex
|   |   |-- plots           # single plots
|   |-- config_data         # where we store feeding zones, area coordinates, calibration, etc. once generated
|   |-- results             # folder we the results of metrics are stored
|   |   |-- feeding         # feeding metrics
|   |   |-- <metric>        # each metric has its own folder
```

Developer Notes

Linting and Style Checking

Use `flake8` to check the code style and linting. Install `flake8` with `pip install flake8` and run `flake8` in the root directory of the project. The configuration file is `.flake8`. Use `black path_to_file.py` to format the code.

Installation inside other projects

Installing the `fishproviz`-module inside other projects requires specifying the respective input-data beforehand in fishproviz/config.env. For that, please refer to 1. Trajectory Visualization > Configuration Afterwards, build the fishproviz-package using the following command:

```
python3 setup.py install
```

This installs the fishproviz-package inside the current python-environment, which needs to match the target project's environment.

PDF-Generation of the Readme.md file

Use Pandoc to generate a README.pdf:

```
pandoc README.md -o README.pdf
```

Caution with `config` module (for developers)

1. The `config` module load the `config.env` file and makes the variables available as global variables. When importing with variables with `from config import var` the `var` will not whiteness and updates to `config`, therefore its recommended to `import config` and access the variables with `config.var`.
2. In `utils.transformations` we have two dictionaries `global FUNCS_PX2CM, global AREA_FUNCS` with function to return the area and calibration data, respectively, give a `camera_position` string. When the area_config is updated these global variables need to be set to `None` such that the new data is loaded.

Possible Future Work

- latex scripts are slow on the PC of the lab, suspicion that the `ls` search is slow.
- feeding-shapes: In the future we want to store the feeding zones in the `path_csv_local` folder and not with the recordings.

Useful Git Routines

- `git pull` to pull the changes from the remote repository
- `git checkout filename` to discard the changes in the file
- `git status` to check the status of the repository
- `git add .` to add all files to the staging area
- `git commit -m "commit message"` to commit the changes
- `git push` to push the changes to the remote repository