1 Software Documentation

The python codes for our assay can be found on GitHub: `https://github.com/GershowLab/TrainingChamber`. The following code is used in our experiments:

- **livetracker.py**: the main software; this code is executed to run experiments and contains functions for tracking the larva, initializing the camera and GPIO outputs, background initialization, and recording the larva's behavior; this code reads in scheme_longterm_template.txt to determine experimental parameters
- **scheme_longterm_template.txt**: a text file where the experimenter inputs training scheme parameters
- **lightsvalvesobjects.py**: contains functions for the training schemes and for changing airflow based on the larva's location
- **statemachine.py**: creates the state machine and defines the seven states: the center of the maze, the 3 channels, and the 3 circles (see Figure 1).
- **region.py**: creates the Region class; contains functions for processing the location of the larva and determining the larva's state from it's location
- **BakCreator.py**: includes functions for creating and updating the background (see Figure 2).

Our Python software uses the open source computer vision software library OpenCV, and runs in a Python virtual environemnt. Prior to running experiments, install OpenCV on your computer. This installation process should also include the creation of your virtual environment; we use 'virtualenv' and 'virtualenvwrapper'. Add the following to the bottom of your `/.profile` file:

```
export WORKON_HOME=\$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

You also want to create the Python virtual environment by entering the following into the terminal:

```
mkvirtualenv cv -p python3
```

The above steps only need to be done once, as part of the OpenCV installation process.

2 Experiment Protocol

Ensure sure you're in the 'cv' environment by entering the following into the terminal:

```
source ~/.profile
workon cv
```

Open the `scheme_longterm.txt` file. This is the only file the experimenter will need to edit to run one of the existing training schemes! To create your own training scheme, see below. The experimenter controls the following parameters through this file:

- **Setup Name**: identifier for the apparatus
- **Date**: experiment date
- **Morning or Evening**: if running more than one experiment, enter 'AM' or 'PM'

- **Training Number**: the number of training cycles for the experiment
- **NaiveTime**: the time (in seconds) the experimenter wants for the pre-training testing period
- **Testing Time**: the time (in seconds) the experimenter wants for the post-training testing period
- **If reward only part of training - how many reps rewarded?**: if the experimenter wants to only pair reward to a certain number of training cycles (e.g. for extinction experiments), enter that number here.
- **Reward only part of training - when?**: if rewarding only during part of the training, enter:
  - **A**: to reward all cycles
  - **F**: to reward only the first N cycles
  - **L**: to reward only the last N cycles
  - **B**: to reward N cycles, present extinction cycles, then to present N reward cycles again
- **# of Block Repetitions**: if experiment wants to do more than 1 training/testing block, enter that number here.
- **Training Type**: select the training type here:
  - **N**: paired training, where reward onset occurs at exactly the presentation of the odor
  - **R**: reverse-paired training, where reward onset occurs at exactly the removal of the odor
  - **P**: phase-shifted training, where reward onset is offset either before or after the odor presentation
  - **L**: training with longer air spacing, where the timing between reward/odor presentations is extended
  - **S**: spaced training, where training follows a spaced protocol (15 minute interval between training)
- **Phase Shifted Training**: if choosing to do a phase-shifted training protocol, select the offset here:
  - **CO2 First**: reward is offset after odor presentation
  - **LightFirst**: reward is offset before odor presentation
- **Longer air training**: if choosing to do the extended-spacing training, select the protocol here:
  - **Paired**: 15 s CO_2 presentation is followed by 15 s reward, then 45 seconds of air only.
  - **Unpaired**: 15 s reward is followed by 15 s CO_2 presentation, then 45 seconds of air only.
  - **DoubleCO2**: 15 s CO_2 presentation followed by 15 s reward, then another 15 s CO_2 presentation; then 30 s of air only.
- **CO2 off during training**: to turn off odor during the training, select 'Y'; else select 'N' (by default, should always be set as 'N').
- **Spaced Training: number of rewards per training**: if choosing a spaced training protocol, select how many training cycles will be presented during each training.
- **Spaced Training: number of repetitions**: if choosing a spaced training protocol, select how many times you would like the spaced training to be repeated; there is a 15 minute gap between each repetition.

After the larva has been loaded into the Y-maze, open the `livetracker` program in the terminal:

```
python livetracker.py
```

The camera will open, and the user will be prompted to select the states for the state machine. The states are selected as follows:

- Intersection: state 1
- Channel 1: state 2
- Channel 2: state 3
- Channel 3: state 4
- Circle 1: state 5
- Circle 2: state 6
- Circle 3: state 7

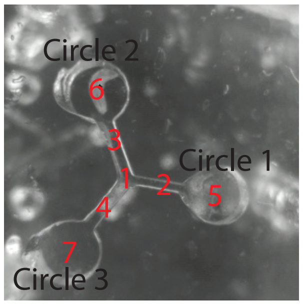

Figure 1: **The seven states in the state machine.** When initializing the software, the user selects the seven states, corresponding to the center intersection (1), the 3 channels (2-4), and the 3 circles (5-7).

Please note that if your camera is oriented differently, you may be selecting the states in a different orientation.

After the user has selected the states, the software will run without any additional input from the user!

The software takes one minute to build the background. The background is the median of all frames collected over the course of one minute; because the larva is moving throughout the maze, it is removed from the otherwise static background when finding the median image. The software updates the background image by adding the current frame to the median image computation if more than 2% of the pixels in the background change (Figure 2B).

After the background collection, the experiment begins. The larva will be tracked, and depending on the larva's state, valves will open/close to direct airflow. This is specified in the Methods section **Behavioral experiments**. The experimenter can view the camera output, with a dot overlaid indicating the tracking position.

To track the larva, at each frame the software first subtracts the static background from the camera frame. The background subtracted image (Figure 2C) segments the larva from the background maze. The image is then thresholded by the program. Thresholding is the software's conversion of a grayscale image to a binary image, in which all pixel values are changed to either 0 or 255. In the thresholded image, the eliminated background pixels are set to 0, while the remaining pixels (the larva) are set to 255.

As a result, we separate the larva from the background in a way that the software can easily detect (Figure 2D). The software locates the larva in the processed frame in real time by finding the largest contour in the thresholded frame. Our tracker consistently detects the location and motion of the larva, since the larva is the only moving part of the environment. The software tracks the center of the larva contour, as shown by the red dot overlaid on the image in Figure 2A.

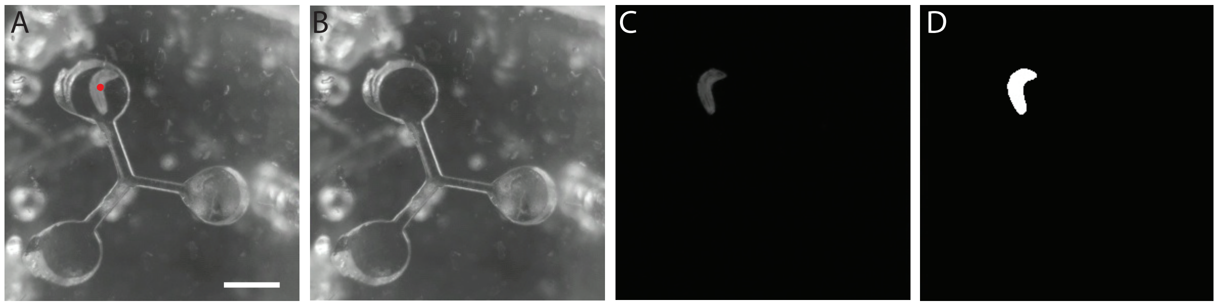

Figure 2: **Custom computer vision software detects the motion of the larva.** The location of the larva is tracked in real time. Computer controlled valves manipulate the direction of airflow in the maze depending on the location of the larva. (A) The camera frame. This image is processed by our software to locate the larva. The red dot superimposed on the image is the location tracked by the software. (B) The background image. The background is the median of all frames collected during the first minute of the experiment; since the larva is moving throughout the maze, the larva is removed from the background when finding the median image. (C) The background subtracted image. The software first processes the camera frame by subtracting the background image from the frame. The result is a frame which segments the larva from the background. (D) The thresholded image. The software converts the grayscale image in (C) into a binary image. The software tracks the larva by finding the largest contour in the thresholded frame. Since the larva is the only changing part of the environment, the tracker reliably detects the larva. Scale bar = 1 mm.

After the first hour of testing, training occurs, using the protocol the experimenter input in the `scheme_longterm.txt` file. After training, the program re-collects the background and tracks the larva as detailed above.

After the experiment has completed, the program will save the following files:

- Text file containing detailed list of all the decisions made by the larva during the pre-training hour, including the direction of airflow in the assay.
- Text file containing only a binary list of the larva's choices during the pre-training hour, with 1 representing a choice of the CO_2-containing channel, and 0 representing a choice of the pure air channel.
- Text file containing the larva's position in the maze at every frame during the pre-training hour.
- Text file containing detailed list of all the decisions made by the larva during the post-training hour, including the direction of airflow in the assay.
- Text file containing only a binary list of the larva's choices during the post-training hour, with 1 representing a choice of the CO_2-containing channel, and 0 representing a choice of the pure air channel.
- Text file containing the larva's position in the maze at every frame during the post-training hour.
- Video file (.h264) of larva during pre-training hour.
- Video file (.h264) of larva during post-training hour.

2.1 Creating a new training protocol

If the experimenter would like to add a new training protocol or modify an existing training protocol, they can do so in the `lightsvalvesobject.py` file. Each existing training protocol is a function in this Python file.

To create a new training protocol, define a new function with the protocol specified in the `lightsvalvesobject.py` file. For example, let's call the new training protocol 'usertraining', with an identifier 'U'.

To make that protocol an option to run, add the following to `Training Protocols` in the `livetracker.py` file:

```
if trainingtype == "U":
    print("User Training: Protocol")
    lightsvalves.usertraining(ptnumber)
```

In the `scheme_longterm.txt` file, simply enter 'U' in the Training Type section, and this protocol will run!