DANCE Analysis Workflow - README

Overview

This document provides step-by-step instructions for processing and analyzing JAABA classifier score files, extracting behavioral data, and visualizing results.

Prerequisites

I. Install Required Packages

1. Download and install Python (recommended: 3.8+); Use Visual Studio Code (VS Code) or another preferred IDE for execution
2. Install dependencies using:
   - pip install pandas numpy openpyxl matplotlib

     For Conda users:

   - conda install pandas numpy openpyxl matplotlib
3. Download and install MATLAB
4. Janelia Animal Automatic Behavior Annotator (JAABA)
   - Download and install: https://sourceforge.net/projects/jaaba/files/
   - JAABA Git repository: https://github.com/kristinbranson/JAABA
   - JAABA documentation: https://jaaba.sourceforge.net/
5. FlyTracker
   - GitHub repository: https://github.com/kristinbranson/FlyTracker
   - Download and install all the necessary files.
6. After installing JAABA and FlyTracker, open MATLAB, go to **Home > Set Path > Add with Subfolders, select both folders, click Save, and then Close.'**

II. Data Organization

Before Analysis

The raw data should be structured as follows:

project_directory/

| -- DD/MM/YYYY/ # Folder named with the recording date

```
│   │-- video1.mp4          # Original experimental movie 1
│   │-- video2.mp4          # Original experimental movie 2
│   │-- video3.mp4          # Original experimental movie 3
```

After Analysis

Once analysis is completed, the data structure will be:

```
project_directory/
│-- DD/MM/YYYY/                  # Folder named with the recording date
│   │-- video1/                  # Original video folder
│   │   │-- video1_0/            # Cropped version 1
          -- video1_0_JAABA/     # Tracker folder for each cropped video
│   │   │   │   │-- perframe/    # Per-frame analysis
│   │   │   │   │-- trx.mat      # Tracking data
│   │   │   │   │-- score_files/ # Post-tracking score files
│   │   │-- video1_1/            # Cropped version 2
│   │   │-- video1_2/            # Cropped version 3
```

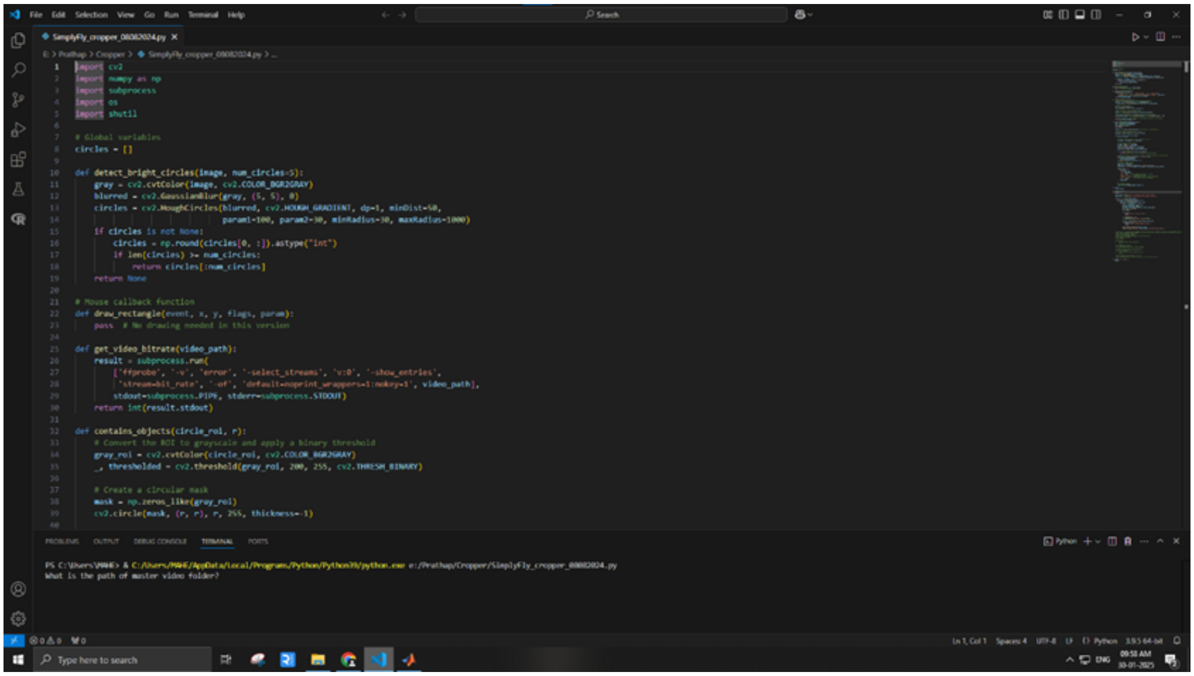
Step-by-step instructions to quantify behavior:

1. Video recording

- Record videos in either .mp4 or .avi format.

2. Video segmentation using cropper script (see README.txt for SimplyFly_cropper_v4.py)

- Double-click on **cropper.py**. This will open the script in your Python IDE (we used Visual Studio code as our IDE).



```python
import cv2
import numpy as np
import subprocess
import os
import shutil

# Global variables
circles = []

def detect_bright_circles(image, num_circles=5):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5, 5), 0)
    circles = cv2.HoughCircles(blurred, cv2.HOUGH_GRADIENT, dp=1, minDist=50,
                               param1=100, param2=30, minRadius=30, maxRadius=1000)
    if circles is not None:
        circles = np.round(circles[0, :]).astype("int")
        if len(circles) >= num_circles:
            return circles[:num_circles]
    return None

# Mouse callback function
def draw_rectangle(event, x, y, flags, param):
    pass  # No drawing needed in this version

def get_video_bitrate(video_path):
    result = subprocess.run(
        ['ffprobe', '-v', 'error', '-select_streams', 'v:0', '-show_entries',
         'stream=bit_rate', '-of', 'default=noprint_wrappers=1:nokey=1', video_path],
        stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
    return int(result.stdout)

def contains_objects(circle_roi, r):
    # Convert the ROI to grayscale and apply a binary threshold
    gray_roi = cv2.cvtColor(circle_roi, cv2.COLOR_BGR2GRAY)
    _, thresholded = cv2.threshold(gray_roi, 200, 255, cv2.THRESH_BINARY)

    # Create a circular mask
    mask = np.zeros_like(gray_roi)
    cv2.circle(mask, (r, r), r, 255, thickness=-1)
```

```
PS C:\Users\MAHE> & C:/Users/MAHE/AppData/Local/Programs/Python/Python39/python.exe e:/Prathap/Cropper/SimplyFly_cropper_08082024.py
What is the path of master video folder?
```

- In **Visual Studio Code**, click on the **Play** button at the top left to run the Python file.
- The Visual Studio Code terminal will ask: **"What is the path of the master video folder?"**
- Copy and paste the master video folder path into the terminal and press **Enter**.
- The terminal will ask: **"How many chambers are there per video?"**
- Enter the number of chambers in the arena and press **enter**.
- The script will process and segment the video into individual folders (as FlyTracker requires video folders, not individual video files).
- When segmentation is complete, the terminal will display: **"Cropped videos for X circles have been saved."** This confirms successful segmentation.

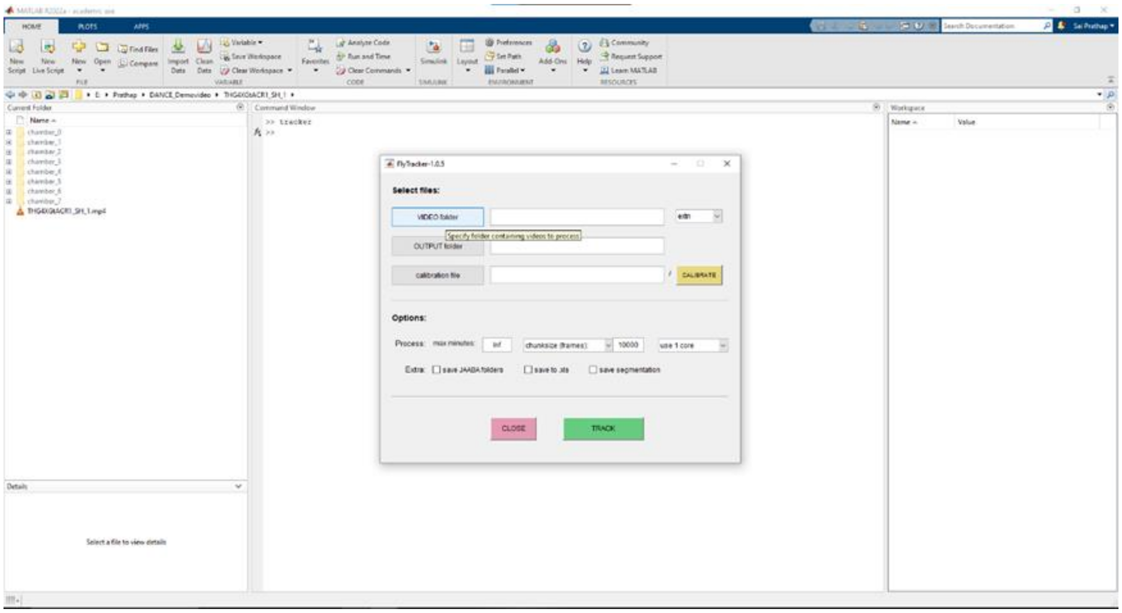
Notes

- Ensure the **master video folder** contains the necessary input video files before running the script.
- The **output folder** will be structured as required for FlyTracker.
- If errors occur, verify that the correct folder path and number of chambers were provided.

3. Fly tracking using MATLAB and FlyTracker

- Open **MATLAB**.
- Click on **Set Path** > **Add with Subfolders** to include both **FlyTracker** and **JAABA** folders.
- In the MATLAB command prompt, type: ‘tracker’

- A dialog box will appear. Click on **Video Folder**, select the appropriate folder, and click **Select Folder**.



```
>> tracker
```

FlyTracker-1.0.5

Select files:

Specify folder containing videos to process

VIDEO folder: .avi

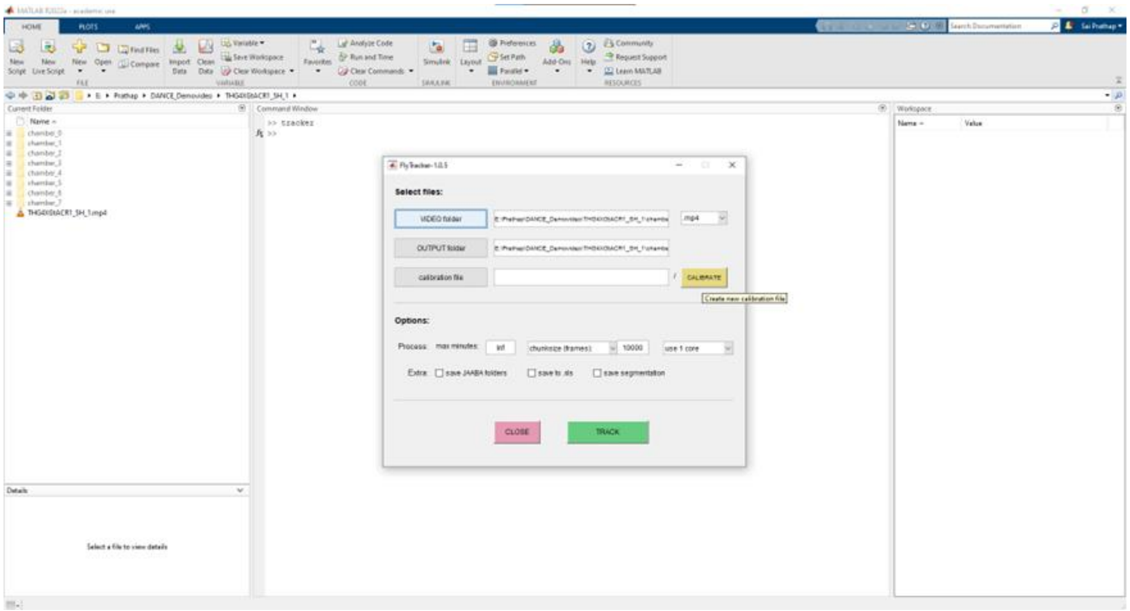
Options:

Process: max minutes: inf   chunksize (frames): 10000   use 1 core

Extra: □ save JAABA folders □ save to .xls □ save segmentation

4. Calibration in FlyTracker (see FlyTracker: https://kristinbranson.github.io/FlyTracker/)

- Click **Calibrate**. A dialog box titled **"Loading Calibrator"** will appear.



```
>> tracker
```

FlyTracker-1.0.5

Select files:

VIDEO folder: E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamb   .mp4

OUTPUT folder: E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamb

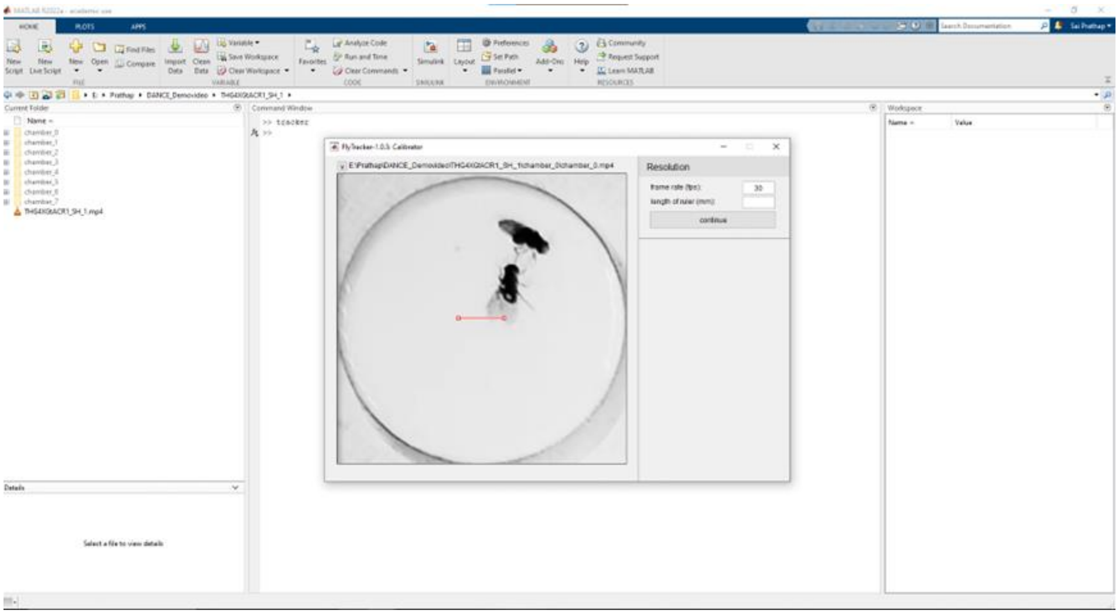
Create new calibration file

Options:

Process: max minutes: inf   chunksize (frames): 10000   use 1 core

Extra: □ save JAABA folders □ save to .xls □ save segmentation

- A chamber outline with red markers will be displayed.



```
>> tracker
```

FlyTracker-1.0.5: Calibrator

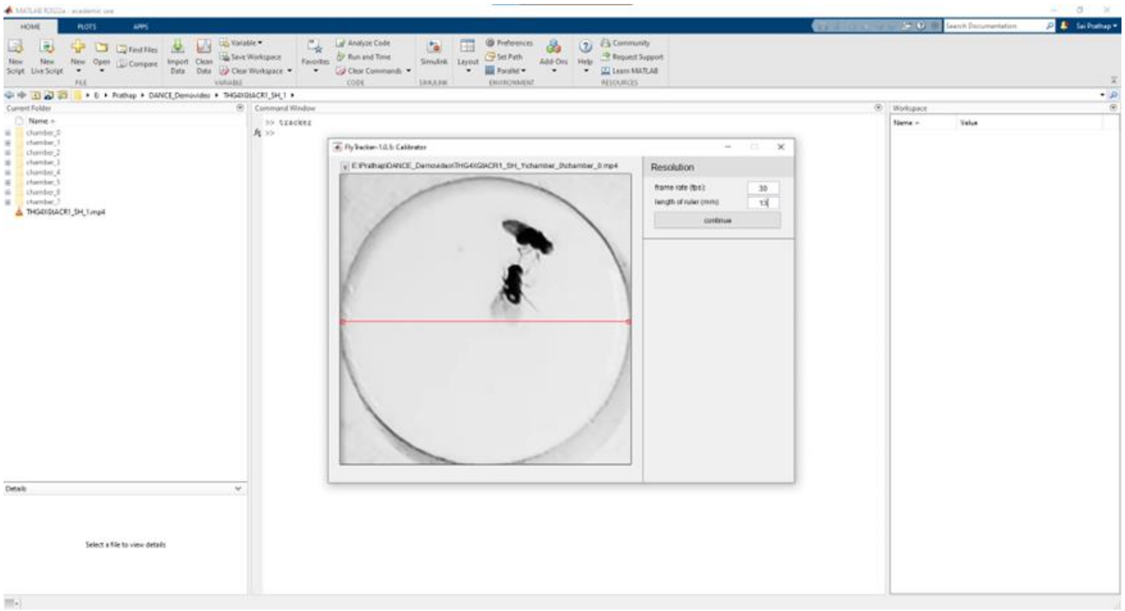
E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_0\chamber_0.mp4

Resolution

frame rate (fps): 30

length of ruler (mm):

- Click and drag the red markers to align with the chamber’s right and left boundaries.
- In the left panel under **Resolution**, enter 13 in the **Length of the ruler (mm)** field (adjust as per the chamber size).
- Click **Continue**.



```
>> tracker
```

FlyTracker-1.0.5: Calibrator

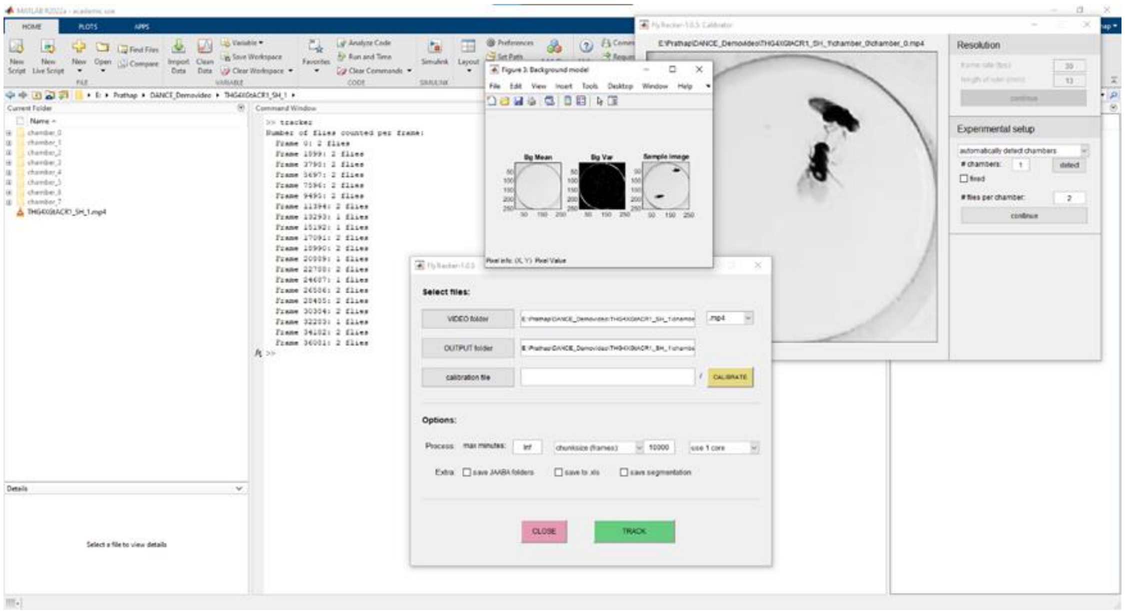
E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_0\chamber_0.mp4

Resolution

frame rate (fps): 30

length of ruler (mm): 13

- A **Progress** dialog box will appear, displaying **"Computing background image"**.
- MATLAB will indicate progress by displaying frame numbers (e.g., 20 minutes × 30 fps = 36,000 frames).
- Once completed, a **Background** model with Bg Mean, Bg Var, and a Sample Image will be shown.



```
>> tracker
Number of flies counted per frame:
  Frame 0: 2 flies
  Frame 1899: 2 flies
  Frame 3798: 2 flies
  Frame 5697: 2 flies
  Frame 7596: 2 flies
  Frame 9495: 2 flies
  Frame 11394: 2 flies
  Frame 13293: 1 flies
  Frame 15192: 1 flies
  Frame 17091: 2 flies
  Frame 18990: 2 flies
  Frame 20889: 1 flies
  Frame 22788: 2 flies
  Frame 24687: 1 flies
  Frame 26586: 2 flies
  Frame 28485: 2 flies
  Frame 30384: 2 flies
  Frame 32283: 1 flies
  Frame 34182: 2 flies
  Frame 36081: 2 flies
>>
```

E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_0\chamber_0.mp4

Resolution
frame rate (fps): 30
length of ruler (mm): 13

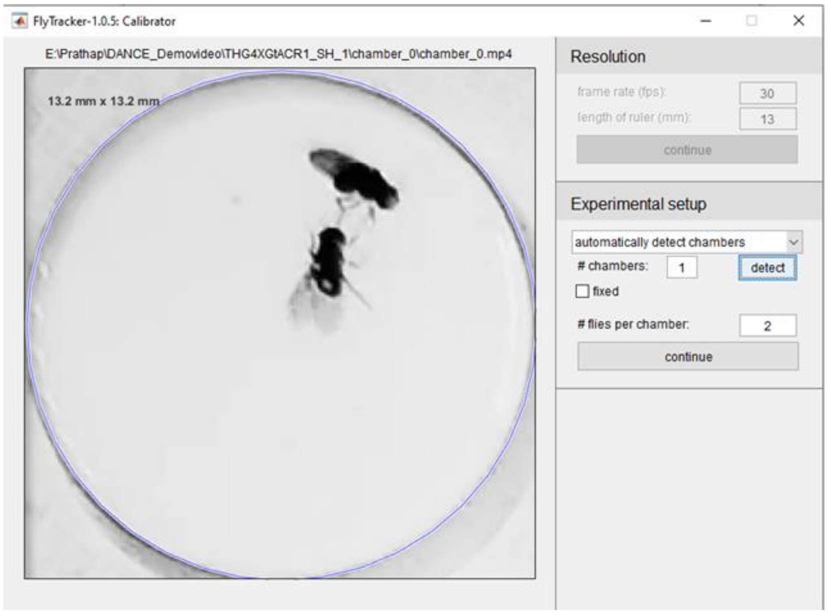
Experimental setup
automatically detect chambers
# chambers: 1
□ fixed
# flies per chamber: 2

Select files:
VIDEO folder: E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chambe .mp4
OUTPUT folder: E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chambe
calibration file:

Options:
Process: max minutes: Inf chunksize (frames) 10000 use 1 core
Extra: □ save JAABA folders □ save to xls □ save segmentation

5. Experimental setup in FlyTracker

- Under **Experimental Setup**, specify the number of chambers and the number of flies per chamber.
- Enter the number of chambers (e.g., 1 or 12) and click **Detect**.
- A **Progress** dialog box will appear, showing **"Detecting chambers"**, with blue circles marking detected chambers.
- If chambers are not properly detected: Click on the **Drop-down menu** under **"Automatically Detect Chamber"**.
- Select **"Manually Set Chambers"** and adjust chamber boundaries accordingly.
- The system will automatically detect **# Flies Per Chamber**.
- Click **Continue**.



FlyTracker-1.0.5: Calibrator

E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_0\chamber_0.mp4

13.2 mm x 13.2 mm

Resolution

frame rate (fps): 30

length of ruler (mm): 13

Experimental setup

automatically detect chambers

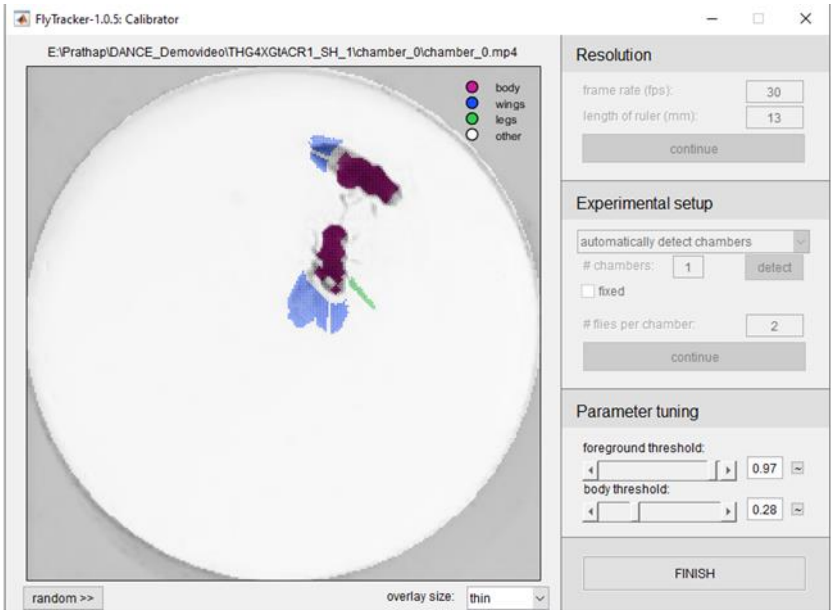
chambers: 1

□ fixed

flies per chamber: 2

6. Parameter Tuning and Tracking

- Adjust the **Foreground** and **Body Threshold** to accurately identify fly body, legs, and wings using the color representation at the chamber's corner.
- Click **Finish**.



FlyTracker-1.0.5: Calibrator

E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_0\chamber_0.mp4

Resolution

frame rate (fps): 30

length of ruler (mm): 13

Experimental setup

automatically detect chambers

chambers: 1

□ fixed

flies per chamber: 2

Parameter tuning

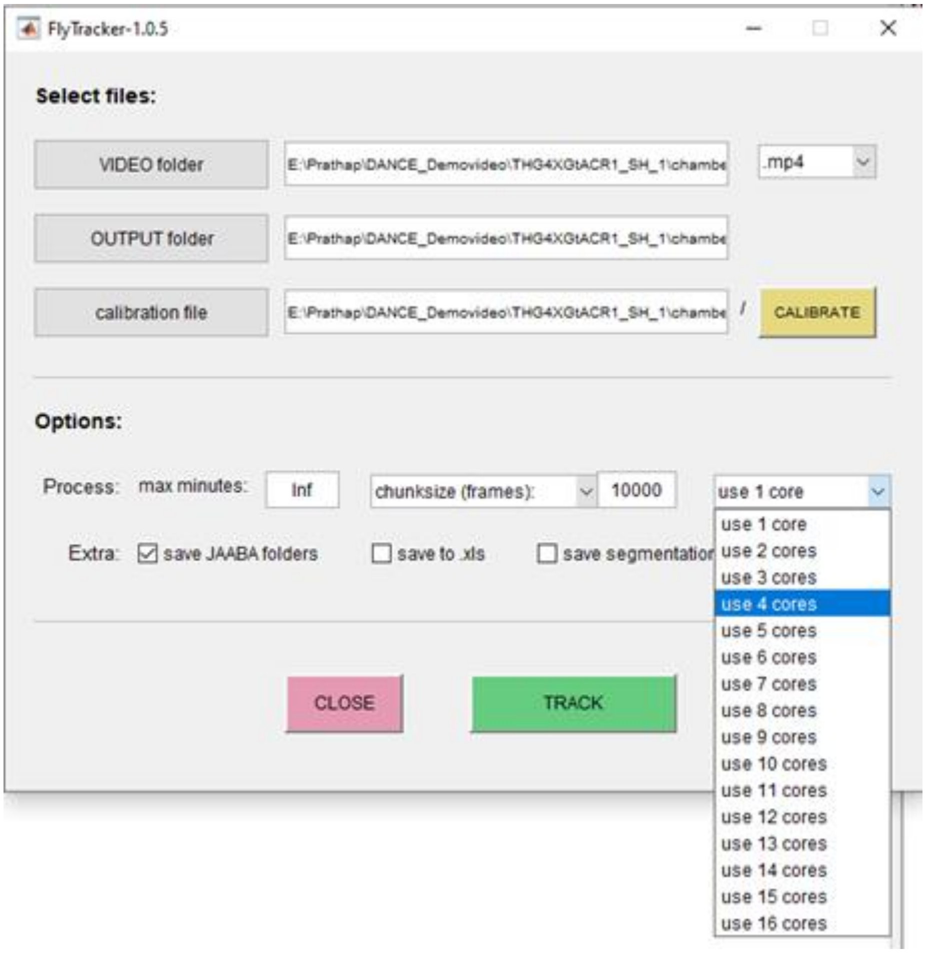
foreground threshold: 0.97

body threshold: 0.28

overlay size: thin

- A dialog box will appear: **"Finalizing Calibration"**.
- Enter the number of CPU cores for computation.
- Click **Save JAABA Folders**.
- Click **TRACK** to begin tracking.

- Once completed, a JAABA folder will be created inside the master folder, containing a perframe folder with perframe features and a trx.mat file.



FlyTracker-1.0.5

Select files:

VIDEO folder E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chambe .mp4

OUTPUT folder E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chambe

calibration file E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chambe /

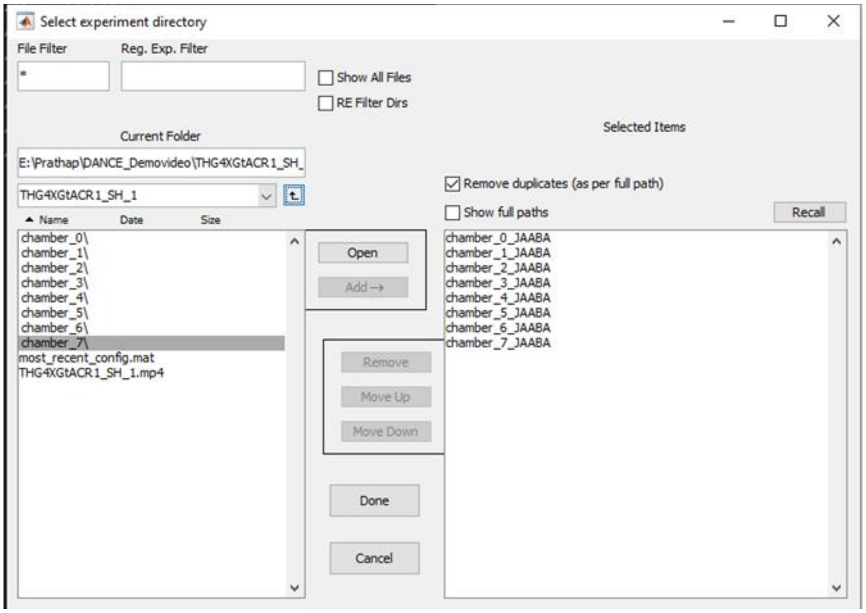
Options:

Process: max minutes: Inf chunksize (frames): 10000 use 1 core

Extra: ☑ save JAABA folders ☐ save to .xls ☐ save segmentatior

7. Generating score files using JAABA

- Open MATLAB and type the following command in the command prompt: JAABAPlot
- A dialog box will appear. Under the **Experiments** section, click **New**, assign a file name (e.g., use the date of the video folder, such as 28112024), and click **OK**.
- A color selection dialog box will appear. Choose color-blind-friendly colors and press **OK**.
- Click **Add**, and a dialog box will appear for selecting the experiment directory.
- Copy and paste the tracked video path into the folder selection window.
- Double-click on the folder, add the **JAABA** subfolder and press **Done**. This step ensures that all JAABA folders are added.



File Filter: *

Current Folder: E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_

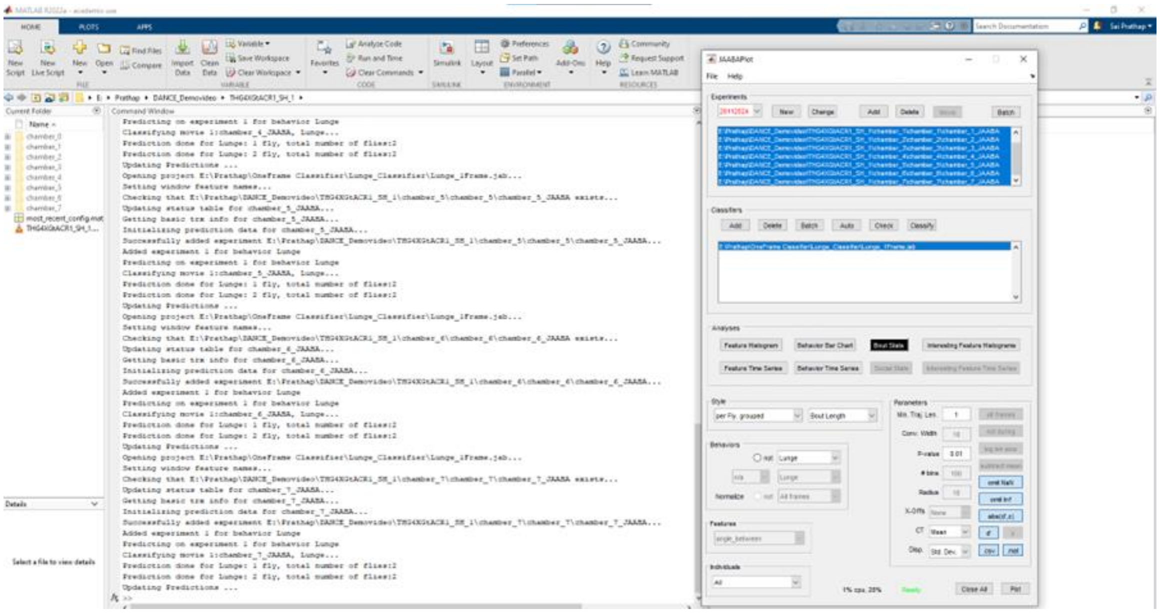
THG4XGtACR1_SH_1

```
chamber_0\
chamber_1\
chamber_2\
chamber_3\
chamber_4\
chamber_5\
chamber_6\
chamber_7\
most_recent_config.mat
THG4XGtACR1_SH_1.mp4
```

Selected Items:

```
chamber_0_JAABA
chamber_1_JAABA
chamber_2_JAABA
chamber_3_JAABA
chamber_4_JAABA
chamber_5_JAABA
chamber_6_JAABA
chamber_7_JAABA
```

- Under the **Classifiers** section, click **Add** and navigate to the **lunge** or **courtship** classifiers to quantify behaviors.
- Click **Classify**. Once processing is complete, it will display **Ready**, confirming the successful generation of score files. These files will be located in the **JAABA** folder within the master video directory, alongside the perframe folder and trx.mat file.



```
Predicting on experiment 1 for behavior Lunge
Classifying movie 1:chamber_4_JAABA, Lunge...
Prediction done for Lunge: 1 fly, total number of flies:2
Prediction done for Lunge: 2 fly, total number of flies:2
Updating Predictions ...
Opening project E:\Prathap\OneFrame Classifier\Lunge_Classifier\Lunge_1Frame.jab...
Setting window feature names...
Checking that E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_5\chamber_5\chamber_5_JAABA exists...
Updating status table for chamber_5_JAABA...
Getting basic trx info for chamber_5_JAABA...
Initializing prediction data for chamber_5_JAABA...
Successfully added experiment E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_5\chamber_5\chamber_5_JAABA...
Added experiment 1 for behavior Lunge
Predicting on experiment 1 for behavior Lunge
Classifying movie 1:chamber_5_JAABA, Lunge...
Prediction done for Lunge: 1 fly, total number of flies:2
Prediction done for Lunge: 2 fly, total number of flies:2
Updating Predictions ...
Opening project E:\Prathap\OneFrame Classifier\Lunge_Classifier\Lunge_1Frame.jab...
Setting window feature names...
Checking that E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_6\chamber_6\chamber_6_JAABA exists...
Updating status table for chamber_6_JAABA...
Getting basic trx info for chamber_6_JAABA...
Initializing prediction data for chamber_6_JAABA...
Successfully added experiment E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_6\chamber_6\chamber_6_JAABA...
Added experiment 1 for behavior Lunge
Predicting on experiment 1 for behavior Lunge
Classifying movie 1:chamber_6_JAABA, Lunge...
Prediction done for Lunge: 1 fly, total number of flies:2
Prediction done for Lunge: 2 fly, total number of flies:2
Updating Predictions ...
Opening project E:\Prathap\OneFrame Classifier\Lunge_Classifier\Lunge_1Frame.jab...
Setting window feature names...
Checking that E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_7\chamber_7\chamber_7_JAABA exists...
Updating status table for chamber_7_JAABA...
Getting basic trx info for chamber_7_JAABA...
Initializing prediction data for chamber_7_JAABA...
Successfully added experiment E:\Prathap\DANCE_Demovideo\THG4XGtACR1_SH_1\chamber_7\chamber_7\chamber_7_JAABA...
Added experiment 1 for behavior Lunge
Predicting on experiment 1 for behavior Lunge
Classifying movie 1:chamber_7_JAABA, Lunge...
Prediction done for Lunge: 1 fly, total number of flies:2
Prediction done for Lunge: 2 fly, total number of flies:2
Updating Predictions ...
>>
```

8. Lunge behavior quantification

- Locate the **Lunge_Analyzer_indiC.py** script (see README.txt).
- Double-click on Lunge_Analyzer_indiC.py to open the script in **Visual Studio Code**.
- In Visual Studio Code, click the **Play** button at the top left to execute the script.

- The VS Code terminal will ask: What is the complete path of the parent folder containing all videos and JAABA folders?
- Copy and paste the full path of the parent directory and press **Enter**.
- The script processes the score files and generates compiled .csv files containing behavior quantification data.

Results and data output

- The final output consists of **compiled score files** in .csv format, which contain quantification of the analyzed behaviors. These results facilitate downstream statistical analysis and visualization.

9. Courtship behavior quantification

Compiles CSV files based on specific classifiers, aggregating courtship behavior data for analysis. IIdentifies male flies by assuming the male’s wing extension counts are higher than the female’s, then assigns that identity to the respective fly for all other behaviors. Generates new CSV files summarizing the data.

- Locate **SheetMaker_postprocessed.m**, add it to the **MATLAB path** using **Set Path > Add with Subfolders**, then click **Save** and **Close**.

Next, open **coutship_bout_num_extractor.py** in **Visual Studio Code**, run the script, and when prompted, enter the complete path of the parent folder containing both **video** and **JAABA** folders. **Results and data outputs:**

- **A folder named "score_files" contains renamed and organized files.**
- **A compiled CSV file for each classifier.**
- **Extracted courtship behavior frames in terms of bouts.**

Alternative: Processing Entire Videos Without Segmentation

You can process the entire video without segmenting chambers by following the same steps for FlyTracker and JAABA analysis.

To quantify courtship bouts in unsegmented videos, use:

coutship_bout_num_extractor_unsegmented.py – This script works just like the previous one and follows the same steps.