

The New Ant Tracker Documentation

Brief Introduction

- The tracking software is heavily based on Srini Ananthakrishnan's [Kalman Filter Multi-Object Tracking Repo](#).

Track.py

- Tracking starts with **track.py**
- **track.py** parses all the arguments provided by the user in **config.yaml** to run the tracker.
- As seen in the **snakefile**, **track.py** accepts many command-line arguments:

```
rule track:
    input:
        track_input
    output:
        'intermediate/track/{video}/{split}/ROI_{roi}.csv'
    threads: 32
    shell:
        'python scripts/track.py {{input}} {{output}} -m {} -c {} -g {} -it {} -d {}'
        '-cto {} -ctt {} -cas {} -tt {} -dm {} -tdt {} -ttl {} -nac {} -eb {}' \
        .format(*(config['tracks'][x]
                    for x in ['min-blob', 'count-warning-threshold',
                             'num-gaussians', 'invisible-threshold', 'min-duration',
                             'canny-threshold-one', 'canny-threshold-two',
                             'canny-aperture-size', 'thresholding-threshold',
                             'dilating-matrix', 'tracker-distance-threshold',
                             'tracker-trace-length', 'no-ant-counter-frames-total',
                             'edge-border']))
```

- If you want to change any command-line arguments, change them in the **config.yaml** file and not the constants.py file.
 - **constants.py** is only useful when running the pipeline locally or without the use of the **config.yaml** file
- The whole point of **track.py** is to parse all the arguments and then call *trackOneClip*, a function in **track_one_clip.py**.

Track_one_clip.py

- The tracking software depends heavily on OpenCV (Open Computer Vision)
- The **trackOneClip** function creates the *detector* and *tracker* objects
 - *detector*: Detects the ants on the current frame of the video. It does not attempt to categorize its detections. It returns the centers of each ant detected at that frame (centers being the center of mass of the ant). (more details later)

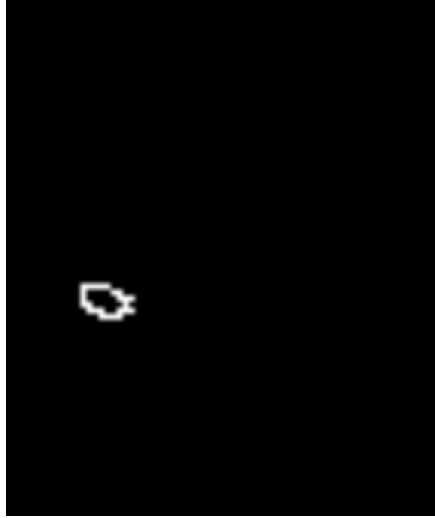
- *tracker*: Takes the centers returned by *detector* and tracks the ants by matching up detections across frames. (more details later)
- The **trackOneClip** function returns the *tracker* object, which contains all of the ant tracks that were detected throughout the clip (basically what is important is the list of histories inside the *tracker* object, we will discuss this later).
- **trackOneClip.py** also contains the **make_history_CSV** function. This function takes the *tracker* object outputted by the **trackOneClip** function as well as an output path. It then makes the track CSV file (contained in `intermediate/track`), which contains information about every ant track that was detected even those which are clearly not ants.
- **trackOneClip.py** also contains the **make_merge_vids** function. This function reads the track CSV created by the **make_history_CSV** function. Some rows will have data in the “merge_id, merge_time, unmerge_id, unmerge_time” columns. This function will get these times and trim the video such that the output will only contain parts of the video where a supposed merger and/or unmerger has occurred
 - There will be two video outputs, one with annotations and one without.

Detector.py

- **detector.py** takes as input a frame of a video (of an ROI, or Region of Interest). It is its own python file and it contains the *Detector* class. It is called by **track_one_clip.py** and creates a *Detector* object.
 - Example:



-
- Then, it does a multitude of steps (grayscale, blurring, edge detection, etc.), such that the ant is the only thing that is pronounced



-
- We then can confidently identify each center for this one frame. We also detect the calculate the area of pixels contained within the edges. We then return both the centers (as a list of coordinates for each center) and the corresponding area.

Tracker.py

- **tracker.py** contains three classes: **History**, **Active_Track**, and **Tracker**
- Both **History** and **Active_Track** only contain data → ie they have no functions associated with them
- **Tracker** class: tracks the ants in a given frame
 - Object **Initialization**: Creates a list of **Active_Tracks** and **Histories** ← will be explained later
 - The list of **Histories** will be used to make the csv file with the function **make_history_CSV**.
 - **copy_track_to_history** static function: copies the information from an **Active_Track** to its corresponding **History** (how to find corresponding **Active_Track** to **History** will be explained later)
 - **Update** function: As input, it takes the coordinates of ants detected by **detector.py** as well their corresponding area
 - Given the coordinates, it then predicts the next coordinate
 - Also with the coordinates, it determines whether or not it is part of an existing track or if a new track has to be created
 - It also detects whether or not a merger/unmerger has occurred
- **Active_Track** class: records real-time data of an ant track that is currently on the frame or has just recently left the track. Once the ant has left the frame for a while (i.e. it's no longer active), its associated **Active_Track** object will be deleted.
- **History** class: records real-time data of an ant track. But unlike **Active_Track**, the object will never be deleted, even after the ant is no longer active on the frame.

- Important: each ant is given an ID when they are first seen on the frame. This ID corresponds to the element index in the *histories* list in the **Tracker** class.