anTraX

Table of Contents

Classifying tracklets

Graph propagation

Validating the tracking results

Analyzing tracking results

Using DeepLabCut for analyzing individual behavior

Using anTraX on HPC environment

Tips and best practices

trails

anTraX - Tracking color tagged ants

anTraX is a software for video tracking ants tagged with a unique pattern of color dots. It was designed for behavioral experiment using the Clonal Raider Ant *Ooceraea biroi*, but can be used for any other model system. anTraX is a **brute force** type tracking algorithm, which was designed to handle high throuput long duration experiments (many colonies over many days). Therefore, it will require considerable computational resources.

The software was designed and written by Jonathan Saragosti and Asaf Gal of the Laboratory of Social Evolution and Behavior in the Rockefeller University, and is distributed under the GPLv3 licence.

References

TBA

Requirements

anTraX works natively on machines running Linux or OSX operating system. It benefits significantly from a multicore system. It is recommended to have at least 2GB of RAM per used core, and a similar sized swap. Computational GPU will speedup the classification phase considerabley.

Python

anTraX requires Python version 3.6 or above. It is hughly recommended to install and use anTraX inside a virtual environemnt, using conda or any other environment manager.

MATLAB

anTraX required MATLAB 2019a or above. If you have a licensed MATLAB installed on your machine, you are good to go. Otherwise, install the freely available MATLAB Runtime for version 2019a.

Other dependencies

On Ubuntu:

```
sudo apt install ffmpeg git tk-dev
```

On OSX using homebrew:

```
brew install ffmpeg
```

Clone the repository

Change into your favorite place to install code packages, then clone the anTraX repositpry:

```
git clone http://github.com/Social-Evolution-and-Behavior/anTraX.git
```

Install

```
cd anTraX
pip install .
```

If you are using a full MATLAB installation, add these lines to your `~/.bash_profile` (on OSX) or `.profile` (Linux):

```
export ANTRAX_PATH=<full path to anTraX repository>
export ANTRAX_USE_MCR=0
```

Otherwise, if you are using MCR, add these lines:

```
export ANTRAX_MCR=<full path to MCR installation>
export ANTRAX_PATH=<full path to anTraX repository>
export ANTRAX_USE_MCR=1
```

For the changes to take effect, run:

```
source ~/.bash_profile
```

Add anTraX to your MATLAB path

If you intend to use anTraX within an interactive MATLAB session, add the repository to your search path. In the MATLAB command line:

```
addpath(genpath(<path-to-anTraX>/matlab));
```

To do this automatically each time you launch MATLAB, add these lines to your `start.m` file.

Developer mode

If you use a full MATLAB installation, changes to the MATLAB source code of anTraX will take effect immediately, as it is always run from the repository directory. If you use MCR, you will need to compile your changes for them to take effect.

Changes made to the python source code will take effect only after reinstalling the package. Alternatively, you can install the package in editable mode, for changes to take effect immediately when you make them:

```
cd anTraX
pip install -e .
```

The experiment directory

The input and output of anTraX are organized in an “experiment directory” (`expdir`). Under this directory, the program expect to find a subdirectory named `videos`, containing the input files. Alongside it, a subdirectory for each tracking session will be created, which will store the parameters and results for that session.

Video files

The input files are a sequence of raw videos, of the same colony, recorded sequentially and under the same conditions. anTraX can process any file format and video encoding readable by ffmpeg, however, all videos are expected to have the same file format, codec, and framerate. Moreover, all videos much have the same base name, followed by a file index suffix seperated with a _ character. Optionally, for convinience, the videos can be organized into subdirectories. This can be a useful way to seperate some meaningful partition of the experiment such as periods of consecutive recording, change in experimental conditions or feeding events, or just to partition a very long experiment. The subdirectories should indicate the indexes of the videos stored in them. For example, subdirectory named “1_24” will contain videos 1 to 24.

expdir structure

The frame data files

Optionally, each video will be accompanied by a frame information file, with the same name of the video and with a `.dat` extension. This file should contain a header row with the variable names (e.g. a timestamp, sensor data, etc.) and a value row for each frame in the video. If the file contains a variable named `dt`, it will be interpreted as the precise inter-frame interval, and will be used for tracking instead of the video framerate parameter.

dat file example

An example folder with dataset can be downloaded from this link: LINK

Using antSpy to record data

If your are using *antSpy* to record data, your data will be automatically organized according to these requirements, and no further action will be needed, antSpy is a Linux/Ubuntu app and will work with any FLIR/PointGrey machine vision camera that supports the spinnaker SDK, or any UVC camera. and can record from multiple cameras in parallel.

What is a session?

A tracking *session* is a run of the algorithm with a set of settings and parameters. In the typical case, you will only create one session per experiment. However, sometime it is usefull to play around with a different parameter set without overwriting existing results, or track different parts of the experiment with different parameters sets. In these cases, multiple session will be created. The session, together with its parameters ancd results, is stored as a subdirectory of the experimental directory and is named by the session identifier name.

Launch the anTraX app

To create and configure a tracking session, simply launch the anTraX app by entering the command into a bash terminal (don't forget to activate your virtual/conda environment if using one):

```
antrax configure
```

Any configuration changes are saved on-the-fly. When finished, just exit the app and the session will be saved.

Create/load a tracking session

First, open an experiment by selecting `Open` in the `Experiment` taskbar menu (or choose an experiment you previously worked on the `Recent` menu). If the experiment contains a previously reated session, it will automatically load. Otherwise, you will be promped to create a new one. Once a session is loaded/created, the configuration workflow will appear as tabs in the application window.

You can move between sessions, or create new ones, by using the options in the `Session` menu.

Display video frames

The anTraX application window is divided into two main parts: configuration panel on the left, and the frame viewer on the right. The displayed image will be augmented according to the configuration. The frames in the experiment can be browsed by using the selectors on the top part of the configuration panel, which will appear in most of the configuration tabs. A frame in the experimement can be defined either by its video index (the first selector) and the frame index in that video (the second selector), or by its total index in the experiment (the third selector).

Frame display selection

Create a background image

The first step is to generate a background image.

Create background tab

Use the ***method*** dropdown to select between the possible background computation methods. The ***median*** method computes the background as the per-pixel per-channel median of a set of randomly selected frames. The ***max*** method computes the background as the per-pixel per-channel max value. Select the number of frames for generating a background image. Obviously, the more frames are used the better the background frame is, especially when median method is used. However, 20 will usually give a good tradeoff between computation time and quality. Frames are randomly selected from the frame range.

One BG option will generate a single background to be used throughout the tracking. With this option, you can select a frame range for selecting frames (useful for cases where some parts of the experiment are more suitable for background calculation)

BG per subdir will generate a separate background frame for each subdirectory of videos. This option is useful for cases where there are movements between the location of the arena in the frame between recording session, of some other change in filming conditions.

The ***Create BG*** button will start the background creation process. Depending on the parameters, this might take several minutes. After the computation is done, the new background will be displayed. If several backgrounds are created, you can choose which one is displayed from the BG file dropdown menu.

All background images are saved as png files in the directory `expdir/session/parameters/background/`.

Set the spatial scale

In the second tab, the spatial scale of the videos will be defined. This is required to have all the parameters and results in real world units, which is essential for parameters to be generalized between experiments, and tracking results comparable between experiments.

To set the scale, choose a feature in the image of which the dimensions are known. Choose the appropriate tool from the drop down menu (either ***Circle*** or ***Line***), press the ***Draw*** button, and adjust the tool to fit the feature.

When done, enter the Length/Diameter of the feature om mm in the box, and finish by pressing the ***Done*** button.

Scale tab

Create an ROI mask

The ***ROI Mask*** is used to define the regions of the image in which tracking is performed.

To set a mask, start by either a white mask ("track everywhere") by pushing the ***Reset to White*** or a black mask ("track nowhere") by pushing the ***Reset to Black***. Then, add and remove regions by selecting a tool from the dropdown and drawing on the image. Adjust by dragging the anchor points. When don,e double click the tool. You can repeat this process untill the ROI is ready.

The ***Multi Colony*** option is used to control how a mask with several disconnected ROIs should be treated. If these regions correspond to separate ant colonies. If checked, each of these regions will be treated as a separate colony, containing a full and fixed set of identified ants, and will be saved separately. Use the dropdown to control the numbering order of the ROIs, and the Assign colony labels buttons to manually assign labels to each numbered colony (avoid white spaces in the labels).

The ***Open Boundry*** option is used to mark parts of the ROI perimeter that are "Open" to ants getting in and out of the ROI. This is used to optimize tracking in these regions. Otherwise, the ROI is assumed to be completely closed.

All ROI and colony masks are saved as png files in the directory `expdir/session/parameters/masks/`.

Tune the segmentation

anTrax segment a background substracted image into foreground and background, with the foreground being composed of several connected components ('blobs'). This is a multi parameter process that should be tuned for each experiment.

On the ***Segmentation*** tab, several of the segmentation parameter can be tuned, while displaying the results. The control parameters include:

Segmentation threshold: in units of gray value difference between image and background.

Adaptive threshold: if checked, the threshold will be adjusted locally as a function of the background brightness, causing the segmentation to be more sensitive in darker areas.

Min area (pixels): Blobs below this threshold are discarded.

Closing (pixels): Optional morphological closing, that merge blobs separated by few pixels.

Opening (pixels): Optional morphological opening, eroding thin pixel lines. Min intensity (gray level). Blobs with maximum intensity lower than this value will be discarded.

Convex hull: Fill in the blob convex hull. Using in cases where there is bad contrast between parts of the ant and the background.

Fill holes: Useful when very bright tags are used, that do not have good contrast with the background and appear as 'holes' in the blob.

Min intensity: Optional blob filter, which discard blobs with maximal intensity lower than the threshold value.

The display of the segmented frame can be configured usng the checkboxes below the frame selectors: ROI mask can be turned on/off, blobs can be shown as convex hull curves (default) or as colored segmented regions (better to check the fine details of the segmentation). Text showing the blob area in pixels and maximum intensity value can be displayed.

Image segmentation

Tune single ant size range

anTrax uses the size of individual ant for filtering possible single ant tracklets for classification, and also for calibrating the linking algorithm. The single ant size is defined by the possible size range, which is adjusted in the ***single ant*** tab. For tunning these range parameters, the blobs detected in the displayed frames are marked with green outlines if they are in the single ant range, with red if they are larger, and with pink if they are smaller. It is recommended to scan a decent number of frames throughout the experiment to look for neear-threshold cases. Note that the range doesnt need to perfectly classify blobs, but to capture the possible size range for single ants.

Single ant size range

Tune the linking

TBD

Enter individual tags information

Before classification, you will need to provide the program a list of the ants in the experiment (identified by their color tags). In case your classifier is trained to identify other types of objects (food, brood, prey insect etc.) you will need to provide these as well.

The list of labels must match the the one the classifier is trained with (read more about classifiers).

If your experiment is a multi colony one, it is assumed the ID list is the same for all colonies in the experiment. If it is not the case, give a list that include all possible IDs, and adjust it using a config file as described below.

The label list is defined by the file `expdir/session/parameters/labels.csv`. Each row in the file contains two entries. The first is the label ID, and the second is the category. Three categories exist: ant_labels, noant_labels, and other_labels. The list must include the label 'Unknown' in the other_labels category.

The ***IDs*** tab is an easy way to configure the list of labels for ant marked with two color tags: First, check the boxes of the color tags used. A label list containing all possible combinations will be created. Next, trim the list to include only the actually used combinations. Also add no-ant labels as needed.

Modifying the ID list using config file

Optionally, a configuration file can be written for adjusting the ID list per colony or per time. Currently, the config support remove commands. The file should be text file located in `expdir/session/parameters/ids.cfg`. Each line in the file is interpreted as a command in the format:

```
command colony id from to
```

The time arguments `from` and `to` can be either `start` for the first frame in the experiment, `end` for the last frame in the experiment, `m` followed by a number for the first frame in a movie (e.g.

`m3`), ‘f’ followed by a number for a specific frame in the experiment (e.g. `f4000`), or a combination of a movie and frame for a specific frame in a specific movie (e.g. `m9f1000`).

To remove the id GP for colony C1 for the entire experiment:

```
remove C1 GP start end
```

To remove YY for colony C5 from movie 22 to the end:

```
remove C5 YY m22 end
```

To remove BG for all colonies for frames 20000 to 30000:

```
remove all BG f20000 f30000
```

To remove PP for colony A from frame 100 in movie 4 to frame 2000 in movie 7:

```
remove A PP m4f100 m7f2000
```

Run a batch job

Once the session is configured, the next step will be to run the blob tracking on all the videos in the experiment. To start tracking, simply execute the following command in a terminal:

```
antrax track <experiments>
```

The `experiments` argument can be either a full path to an experimental directory, a full path to a text file with a list of experimental directories (all of which will run in parallel), or a full path to a folder that contains one or more experimental directories (all of which will run in parallel).

The track command accepts the following options:

```
--nw <number of workers>
```

anTraX will parallelize the traking by video. By default, it will use two MATLAB workers. Depending on your machine power, this can be changed by using this option.

```
--movlist <list of movie indices>
```

By default, anTraX will track all movies in the experiment. This can be changed by using this option. Example for valid inputs incluse: `4`, `3,5,6`, `1-5,7`.

```
--session <session name>
```

If your experiment contains more than one configured session, anTraX will run on the last configured one. Use this option to choose a session explicitly.

Checking job progress

Depending on the number and length of video files, tracking jobs can be very long. anTraX will print a report to terminal when a task (tracking of single video) starts/ends. Logs for each task can be found in the experimental directory, under `session/logs/`.

Classification workflow

In this step we will classify each tracklet that was marked as possible single ant tracklet. Each of these tracklet will be assigned as either:

- An ant ID from the list of possible IDs
- A non-ant tracklet, wither as a general category or a specific one if such exist in the classifier.
- A multi ant tracklet
- An ambigious tracklet ('Unknown') in case the classifier couldn't make a decision.

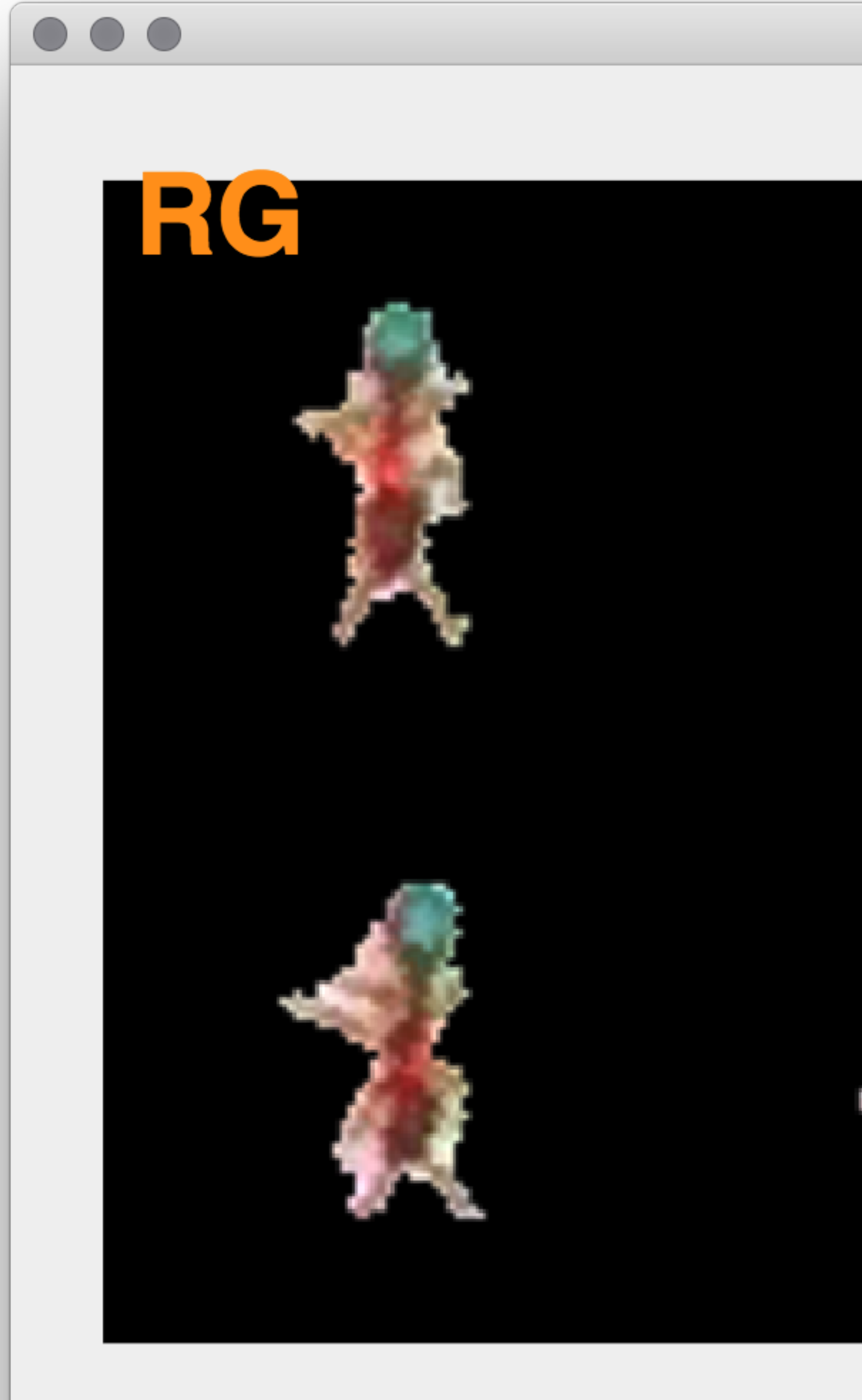
Each tracklet is classified by first classifying all blob images belonging to that tracklet, using the *blob classifier* and then weighting these classifications to produce a whole tracklet classification. The blob classifier is a trained deep convolutional network (CNN), that needs to be trained on a trainset of pre-classified blob images

Creating a training set

The blob classifier is a trained deep convolutional network (CNN), that needs to be trained on a trainset of pre-classified blob images. anTraX includes an interactive GUI application to prepare such a training set from a tracked experiment:

```
antrax extract-trainset <expdir>
```

Note that it might take a few minutes for the app to prepare the image dataset for labeling, especially if the experiment is long and/or multi colony.


RG

The app will display blobs images from a randomly selected tracklet.

Merging training sets

The exported examples are saved as images in the experimental directory, under `/session/classifier/examples/id/`. When creating a classifier specific to that experiment, training can be done from that directory. But in order to create a general classifier to be used on many experiments, it is necessary to create a trainset that contain examples from a few experiments. This can be done with the command:

```
antrax merge-trainset <source-classdir> <dest-classdir>
```

This will merge all the examples from the source classifier directory (usually `expdir/session/classifier`) into the destination classifier directory. It is recommended to keep multi-experiment classifiers seperate from any specific experimental directory to avoid confusion.

Important: the user is responsible to make sure the label list matches when merging trainsets. Otherwise, problems might arise.

Training the classifier

```
antrax train <classdir>
```

Classifying tracklets

To run classification in batch mode:

```
antrax classify <experiments>
```

The `experiments` argument can be either a full path to an experimental directory, a full path to a text file with a list of experimental directories (all of which will run in parallel), or a full path to a folder that contains one or more experimental directories (all of which will run in parallel).

The classify command accepts the following options:

`–classifier

Explicit path to a classifier (`.h5` file created by the train process). By default, anTraX will use the classifier file that exist in the default location in the experimental directory `expdir/session/classifier/classifier.h5`. If it doesn't exist, an erorr will be raised.

```
--movlist <list of movie indices>
```

By default, anTraX will track all movies in the experiment. This can be changed by using this option. Example for valid inputs incluse: `4`, `3,5,6`, `1-5,7`.

```
--session <session name>
```

If your experiment contains more than one configured session, anTraX will run on the last configured one. Use this option to choose a session explicitly.

Validating classification and retraining

The app will display random classified tracklets from the experiments. If the displayed tracklet was assigned with a valid ant ID, you can mark it as correct, wrong, or false positive (i.e. should not have been assigned with an ID). If the displayed tracklet was assigned with a non-ant label, you can mark it as false negative if it is an ant, or just skip to next to mark it as correct. Note that we don't count correctness in assigning the various non-ant labels here. The stats will be appear in the text window, both as per-tracklet error and per-frame error. Ambiguous tracklets are shown but do not have an error response (saying 'I don't know' is not considered an error).

When a tracklet is misclassified, or marked as unknown, you have the option to add its images as examples in the classifier directory. For doing that, choose the correct label from the dropdown menu. If you wish to export all images of that tracklet (be careful, always check the images first), press the **Export all** button. IF only a subset of frames are good examples, press the **Choose frames** button. A second window will appear:

In that window, frames from the tracklet will be presented one by one. For each, either select **Import as ID** if it is a good example (rule of thumb: if you can say which ant it is by this image alone), select **Import as unknown** if it is definitely an unrecognizable image. If in doubt, skip this frame by pressing **Next**.

Important: if you made an error while marking the tracklet as correct/wrong/etc, you can correct by going back and re-marking the tracklet, and it will fix the stats appropriately. However, if images from a tracklet were exported as examples under the wrong ID/label, it is not enough to re-export them under the correct label. You will have to open the classifier directory in the systems's file explorer, look under the wrong label you exported, and delete the images. It is useful to sort by modification date and look at the most recently modified.

Classification is done by evaluating a trained tensor-flow neural network on each blob image. The classifier 'lives' in a self contained file, and can be shared between experiments. The classifier directory contains a file that stores the neural network topology and parameters in `classdir/model.h5`, and a sudirectory named `examples` that contains a data set of labeled images used for training. The example images are organized into subdirectories named by their labels.

Propagating IDs on tracklet graphs

```
antrax solve <experiments>
```

The `experiments` argument can be either a full path to an experimental directory, a full path to a text file with a list of experimental directories (all of which will run in parallel), or a full path to a folder that contains one or more experimental directories (all of which will run in parallel).

The solve command accepts the following options:

```
--nw <number of workers>
```

anTraX will parallelize the traking by video. By default, it will use two MATLAB workers. Depending on your machine power, this can be changed by using this option.

```
--glist <list of graph indices>
```

By default, anTraX will track all graphs in the experiment. This can be changed by using this option. Graph indices are an enumeration of movie groups according to the options set in the session configuration. Example for valid inputs incluse: `4`, `3,5,6`, `1-5,7`.

```
--session <session name>
```

If your experiment contains more than one configured session, anTraX will run on the last configured one. Use this option to choose a session explicitly.

Using the graph explorer to view and debug ID assigments

TBA

TBA

TBA

Install DeepLabCut

Install the DeepLabCut package into your python environment:

```
pip install deeplabcut
```

Export single ant videos for training

Train a DeepLabCut model

Run

```
antrax dlc <experiments> --cfg <path-to-dlc-cfg-file>
```

The `experiments` argument can be either a full path to an experimental directory, a full path to a text file with a list of experimental directories (all of which will run in parallel), or a full path to a folder that contains one or more experimental directories (all of which will run in parallel).

The required argument `cfg` is the full path to the project file of the trained DeepLabCut model.

The classify command accepts the following options:

```
--movlist <list of movie indices>
```

By default, anTraX will process all movies in the experiment. This can be changed by using this option. Example for valid inputs incluse: `4`, `3,5,6`, `1-5,7`.

```
--session <session name>
```

If your experiment contains more than one configured session, anTraX will run on the last configured one. Use this option to choose a session explicitly.

Loading and analyzing postural data

TBA

Installation

In principle, anTraX installation on an HPC environment is the same as installation on any other Linux machine. The main difference is that typically, you will not have administrator priviliges to intall system-wide packages on the HPC. Luckily, there are not many of those required by anTraX. Also install MATLAB Runtime for version 2019a.

We recommend using a conda environemnt to setup anTraX on HPC, as it enables installation of several packages such as ffmpeg. If some pckages are still missing and are not available in conda, work with your system administrator to find a solution.

If you plan on using DeepLabCut, install it into the python environment, and set it to 'light mode' in `~/.profile`:

```
export DLClight=True
```

anTraX workflow on HPC

As computer clusters do not typically support interactive work, this will need to be done on a PC. An example for a tracking workflow using a computer cluster will be as follows:

1. Prepare the experimental directory/ies on a PC.
2. Configure a tracking session for each experimental directory on the PC.
3. Sync the experimental directories into the HPC environment.
4. Run batch tracking on the HPC (see below).
5. If you have a trained blob classifier, jump to step 7. Otherwise, sync tracking results back to the PC, and create a training set using the interactive interface.
6. Sync again to the HPC, and train the classifier.
7. Run the `classify` and `solve` commands in batch mode on the HPC.
8. Sync the results back to the PC.

It is recommended to use an incremental tool such as `rsync` to speed up data transfer.

Batch run on HPC environment

If you are on an HPC environment, using the SLURM workload manager, you can run each of the batch commands (`track`, `classify`, `solve` and `dlc`) using the `--hpc` flag:

```
antrax <command> <experiments> --hpc --hpc-options <opts>
```

anTraX will then submit a SLURM job for each experiment, each containing a task for each video in the experiment.

The optional `hpc-options` argument can controlled some of the SLURM options and accepts a comma seperated list of some of the following options:

```
throttle=<throttle>
```

Number of tasks to run in parallel for each job.

```
partition=<partition>
```

The partition to run in (otherwise, use the system default)

```
email=<email>
```

Send an email for start/end of each job.

Marking ants

- Choose colors that are easily separated in RGB space. As a rule of thumb, colors that are easily separated by a human eye in the video will result in better tracking (surprise!). The color set Blue, Green, Orange, Pink and Yellow usually results in good accuracy.
- It is important to make sure that all tags are visible to the camera for a large part of the frames, at least when an ant is on the move. For that reason, side-byside tags are usually less preferable than placing tags one behind the other.
- It is not advised to use variable number of tags per ant (for example, using "no tag" as an additional color). As tags are often not visible (e.g. when an ant is grooming, or on its side), this can lead to high error rate (i.e. obscured tag is identified as a no-tag ID).
- It is possible to use symmetric color combinations (e.g. Blue-Green and Green-Blue) in the same experiment. However, note that in that case asymmetry in the ant image is necessary for good classification. For example, for *O. biroi*, a thorax tag and abdomen tag will work nicely, assuming the head and the antennae are breaking the image symmetry (don't paint the head!).

Recording videos

- the program is designed to identify dark ants on a light background. The better the contrast, the easier it is to segment the ants from their background.
- It is recommended to tune the camera, especially the brightness, contrast and saturation levels, so to make sure tag colors are maximally distinguishable. A bit of over saturation is usually advised.
- As the software segment the image using background subtraction, it is important to make sure the image is stable throughout the experiment. Particularly, that there are no camera movement, arena movements and illumination fluctuations.
- Sometimes, it is necessary to pause the experiment and to take care of the experimental colonies. In the case that the arenas cannot be precisely returned to their exact position relative to the camera, or that there was a change in the arena themselves that can results in a changing background, it is recommended to start a new video subdirectory to hold movies. Backgrounds can be created separately for each subdirectory.
- Video resolution should be high enough to identify the color tags. As a rule of thumb a tag size of at least 5x5 pixels is recommended.
- Video framerate should be high enough to make sure ant blobs in consecutive frame overlap considerably.

Computer configuration and parallel execution

- It is recommended to setup a MATLAB parallel profile named `catt` using the MATLAB dialog. The software will look for this profile be default. If it doesn't exist, it will use the default profile. A dedicated profile enables optimizing the settings for tracking runs without changing the default settings for other purposes.

- A tracking job will typically use around 125-150% CPU. Therefore, it is recommended to set the number of workers to about 0.5-0.75 of the number of threads supported by your machine. Also, set the number of threads per worker to two. It is also recommended to have at least 2GB of RAM and 4GB of swap per worker.
- Tracking jobs are usually long. It is advised to use a desktop computer or workstation and not a laptop computer. Make sure the turn off auto sleep on the computer.

Tracking setting and parameters

- What is a good background
- median vs max
- What is a good ROI mask
- Good segmentation
- Why single ant threshold are important

Training and classification

- Choosing good examples for training a classifier is an art. On the one hand, it is a good practice to span the space of possible presentations of the ants, and choose “atypical” images. On the other hand, as this space is huge, we are practically guaranteed to be in an under-sampled regime, so it is better no to use examples in ambiguous areas of the image space. As a rule of thumb, don’t use images that are not easily identifiable for a human.
- If you see a systematic classification error between two IDs, especially if there is no clear visual reason for this error, 99% you have some contamination of the training set. Try checking the example directory of the classified label and look for examples that belong to the actual ant ID. Delete them and retrain the classifier.
- Neural networks are best trained using negative examples. Otherwise, they will produce high rate of false positives in cases where none of the known labels exist. In our case, this is represented by the “Unknown” class (and the “Multi” class if exist). It is important to add as much examples as possible to this class. Good images are those that a human cannot identify at all. Tracklets labeled as “Unknown” are not left for the propagation algorithm to assign.
- Why are some tracklets identified as “Unknown” although they are very obvious to recognize? This might be due to a few possible reasons. Sometimes the training set is contaminated (it is always recommended to check the set every now and then). Sometimes this is due to short tracklets having a more conservative classification threshold. Sometimes it is due to apparently good images being discarded because of some frame filter (try running without frame filters - LINK).

ID propagation

Debugging and fixing the tracking

Working with xy trajectories

-