Technical support for `trex.run`

Fritz Francisco
fritz.francisco@hu-berlin.de

November 23, 2021

1 `trex.run`

`trex.run` is a tracking software developed and maintained by Tristan Walter and is designed to track multiple animals from two dimensional images. It is best applied to simple, standardized laboratory footage where both illumination and background are maintained constant and only the object of interest move.
A good documentation and explanation of the usage cases can be found here:

https://trex.run/docs/contents.html
https://github.com/mooch443/trex

1.1 Installation

Before getting started, a few things need to be set up. The most important components for using `trex.run` are a correctly instantiated `conda` environment and the `.settings` file.

1.1.1 conda

`trex.run` is implemented in C++ and Python. To get started with Python it is recommended to install an environment manager. An environment manager lets you create a virtual box within your system, in which you can install a variety of Python dependencies and packages without disturbing, or interacting with the main system. These environments can be used with Python, R, Java etc. One of the most common package management systems and environment management systems is conda. This can be installed following these instructions (under "Regular Installation"):

conda installation

Once `coda` is successfully installed `trex.run` can be installed! For ease of use it is recommended to navigate to the following website https://trex.run/ and use the command suggested there. This is based on the distribution which is automatically recognized through your browser. Otherwise, detailed installation instructions can be found on the official website and in the corresponding documentations as well. In most cases it should suffice to run the following command from within the terminal, once `conda` was correctly set up:

```
conda create -n tracking -c trexing trex
```

The `conda` environment can then be accessed through a Terminal (Linux, OS) or the Anaconda Prompt (Windows). In both cases the environment is opened, once `conda` was successfully installed by executing the following command:

```
conda activate MyEnvironment
```

Here, `MyEnvironment` corresponds to the specific name of the environment created, which in this case is named `tracking`. Therefore, the corresponding command for the `trex.run` installation would be:

```
conda activate tracking
```

1.1.2 The .settings file

The `.settings` file contains all project specific parameters required for converting and tracking, using `trex.run`. These are set in the `.settings` file, which is a common `.txt` file where the ending is replaced with `.settings`. This file is referenced during all further processes, which is why it needs to be created or copied from existing projects and adjusted to match ones needs prior to using `trex.run`. The settings file should NOT contain any leading # symbol, since this will hinder the functionality of it! An example is shown below:

Example.settings

```
analysis_range = [-1,-1]
auto_minmax_size = false
average_samples = 200
averaging_method = "mode"
approximate_length_minutes = 10
blob_size_ranges = [0.2,3]
color_channel = 1
cam_undistort_vector = [-1.95051879012043l001e-01, 4.970165099425187250e-01,
3.6434950053804513 60e-03, 9.2379467106270803 19e-04, -2.257615797069292718e+00]
cam_matrix = [5.32295068503336 2788e+03, 0,
1.7471592087770916 35e+03, 0, 5.3197525521414063 41e+03, 1.7577973693423475 65e+03, 0, 0, 1]
cam_undistort = true
cam_resolution = [3400,3400]
cam_circle_mask = true
cm_per_pixel = 0.022
crop_offsets = [0,0,0,0]
correct_lumincance = true
enable_live_recording = true
frame_rate = 15
image_invert = false
recognition_enable = true
meta_species = "Poecilia formosa"
max_speed = 10
meta_real_width = 76
output_posture_data = true
output_centered = true
threshold = 14
track_threshold = 14
track_max_individuals = 4
track_max_speed = 10
```

Prior to tracking a conversion step is done (see subsection 1.2 Getting Started) which heavily relies on the parameters `threshold` and `blob_size_ranges`. The functionality and further description can also be found in the official documentation (see below).

For tracking, `max_speed` (maximum expected speed of the objects in cm/s), `track_max_speed` (maximum speed to consider feasible during tracking cm/s), `meta_real_width` (real width of observed image in cm) and `track_max_individuals` (number of expected individuals to track) should be set accordingly. Further, it is vital to incorporate a correct measurement of the parameter `cm_per_pixel`. This is easiest done by measuring a know distance in the field of view (taken from a snapshot of equivalent dimensions/resolution as the original image)

and converting it to cm/px. It is recommended to take a snapshot using VLC and measuring it in pixel values using Fiji/ImageJ. All previously discussed parameters are important since they are used to calculate possible matches for individuals and refine outliers during tracking. If the camera was calibrated calibration parameters (`cam_undistort_vector` and `cam_matrix`) can be set as well. When `cam_undistort` is set to `true` this enables the images then to be undistorted during conversion, which can greatly improve the tracking quality. The camera calibration can be done by using a common calibration checkerboard, with which the required calibration parameters (`cam_undistort_vector` and `cam_matrix`) can be calculated. This is done by using a video recorded from the camera to be calibrated, in which the calibration checkerboard was moved in front of the camera. This video, in turn serves as input to the script found here which results in all camera parameters being calculated and returned. Once all parameters have been set accordingly the `.settings` file should be saved on the local machine. It is important to make sure `trex.run` accesses the correct `.settings` file, since it will otherwise revert to the default settings. This can be checked by looking at the terminal output produced after running any command. There, it is explicitly stated which `.settings` file was used.

For further references on how to correctly set up the `.settings` file please follow the official documentation.

1.2 Getting Started

Before using `trex.run` the `conda` environment should be activated. This is commonly done by running the command:

```
conda activate tracking
```

Following this step, the general workflow is made up of two steps:

1. Conversion of the RAW video material, where most common video formats, such as .avi and .mp4 are supported as input. These are converted to a processed video file (.pv).
2. In a second step the individual objects are tracked, using the .pv file as input

It should be noted that for all of the following steps it is highly recommended to used **absolute paths** (i.e. `C:\Users\lab\Documents\test\config.settings` instead of `.\config.settings`)

Conversion is done using the command:

```
tgrabs
  -i /path/to/input.mp4 # raw video
  -s /path/to/Example.settings
  -output_dir /path/to/output/directory
```

Once the video files are converted and the preliminary results and detections are satisfactory the tracking can be done using:

```
trex
  -i /path/to/converted_input.pv #.pv file
  -s /path/to/Example.settings
  -output_dir /path/to/output/directory
```

Both previously described steps can be done using the General User Interface (GUI) as well and following the descriptions. This is done by running

```
tgrabs
```

or

```
trex
```

Tracking with `trex.run` is done by applying background subtraction, detection blobs and tracking these. In a further step the individual identity can be estimated and maintained by training a deep neural network on the individual data. This way, individuals are recognized in each frame and detections merged, based on their overall similarity. The algorithm bases it's assumptions on the fact that only `track_max_individuals` should be detected in any instance and that these individuals can only move `max_speed`.
To train the network and track the individuals using the visual identification this needs to be set using the GUI (top left Menu button) or while initiating `trex.run` from the command line:

```
trex
   -i /path/to/converted_input.pv
   -s /path/to/Example.settings
   -output_dir /path/to/output/directory
   -auto_train # train network
   -auto_quit # quit once traiend and applied
   -no_window # optional when running without interface
```

Here, the parameters `auto_train`, `auto_quit` and `no_window` are specific to using `trex.run` in the command line interface (CLI/Terminal) and applying batch processing (automatically apply to multiple files). For tracking single individuals or when visual identification is not necessary, setting the flag `enable_live_recording` to true is sufficient. This will already generate and save tracking output during the conversion step. This approach will however lead to faulty identification if more than one individual are observed or if the detections are very unreliable (for example with other moving object in the image).

1.3 Individual Recognition

During this step a artificial neural network is trained to recognize the individuals in the video and assign them with consistent individual identities over time. It is to be mentioned that this step requires a graphical processing unit (GPU) in order to run sufficiently. Further, the `track_max_individuals` flag is important for the the algorithm to estimate the correct number of individuals. These flags can be set directly in the terminal when executing the `trex` command or added to the `.settings` file.

To run the visual identification and tracking from the terminal the following command can be executed:

```
trex -i /path/to/input_file.pv   # This is the .pv file NOT the video
    -s /path/to/my_personalized.settings
    -output_dir /path/to/output/directory/
    -track_max_individuals X   # Number of individuals
    -auto_train
    -auto_quit
```

The same process can also be done using the graphical user interface (GUI). Here, one needs to click on

1) **Menu** > **visual identification**

2) **Menu** > **autocorrect** > **Yes**

3) **Menu** > **autocorrect** > **No**

This should start the process of determining individuals based on pixel values, learning these differences and applying them to consecutive frames to create consistent tracks. This process is time consuming and computationally expensive (CPU & GPU) and will likely take a while.

1.4 Manual Correction

It is recommended as best practice to manually approve all tracking results once these are obtained. This is done by opening the .pv files as when tracking them using `trex`. It is important in this step that the `.settings` file parameters match those used for tracking. The output directory should be set to the location of the `trex` output generated during tracking.

```
trex
  -i /path/to/converted_input.pv
  -s /path/to/Example.settings
  -output_dir /path/to/output/directory
```

Once `trex` is opened in the GUI navigate to the **Menu** > **visual identification**. This should start training the neural network on the detected individuals. However, if the visual identification had already been done and the correct output directory, containing this information, was correctly given by setting the `output_dir` flag the visual identification only needs to be loaded and applied:

1) **Menu** > **visual identification** > **Load weights**

2) **Menu** > **visual identification** > **Apply**

Once the visual identification is applied the identities need to be corrected. The first step is to do this automatically by navigating to **Menu** > **auto correct** > **Yes**. After waiting for this step to finish (% progress is indicated in the bottom left of the image), one must do the same step **Menu** > **auto correct** but now selecting the option **No** . After this the identities indicated as numbers over each fish should be either red, green or white corresponding to the quality and success of tracking (red being worst and green being best).

Once visual identification has been applied the individuals need to be manually approved. This is done by visually following the individuals and checking for identity switches. This means, cases where the identity number is carried over to an individual that is not correct. This commonly happens when individual overlap or briefly disappear. To navigate through all possibly erroneous switches one can use the "**M**" (next) and "**N**" (previous) key on the keyboard. Then the faulty individual is selected by clicking (left mouse button) on it and the correct ID is selected by navigating to **Match** at the top of the GUI. This must be done for all individuals and every erroneous identity switch. Once completed the progress can be saved by using **Menu** > **save state**. It is important to save frequently to limit loss of progress in case of a unexpected break of the software or forgetfulness. The state can be loaded during a later stage using **Menu** > **load state** equivalently.

1.5 Best Practice

For best practice and most convenient workflow it is recommended to follow the following steps:

1. Initially set up `conda` environment once (subsubsection 1.1.1 conda).
2. Copy original `.settings` file and only work with the copy (subsubsection 1.1.2 The .settings file).
3. Always make sure the parameters:
   - `threshold`
   - `blob_size_ranges`
   - `max_speed`
   - `meta_real_width`
   - `track_max_individuals`
   - `cm_per_px`

   are set correctly.
4. Copy all files (.pv and trex output) and only work with copies.
5. Check terminal output. The logged information printed there is **VERY** helpful!
6. Do **NOT** restart visual identification if it has already been done! This will result in losing the progress when not working with copied files.
7. Save frequently and create backups. Especially when working on the manual corrections (subsection 1.4 Manual Correction).

2 Useful commands

As some might be working on linux computers or may encouter a linux command line from time to time it may be usefull to know some basic linux commands. Some of these commands also work within `conda` environments:

`ls` : ["list"] lists all files in the current directory

`cd` : ["change directory"] changes into the directory following the command. If no directory is specified it will change into the basal home directory

`cp` : ["copy"] copy specified files to designated directory. cp is followed by the filenames to be copied and then the designated directory

`mv` : ["move"] moves specified files to designated directory which does not replicate them. mv is followed by the filenames to be moved and then the designate directory

`pwd` : ["print working directory"] shows the current directory (i.e. `dir` in Windows)