TRACKTOR v0.1 - tutorial

Last updated: 05.2018

If you find errors in this tutorial, outdated information and/or identify links that are no longer working, please notify Vivek Hari Sridhar by [email](mailto:vsridhar@orn.mpg.de) or by reporting an issue on [GitHub](https://github.com/vivekhsridhar/tracktor). These issues will be fixed as soon as possible.

N.B. Hyperlinks in the text will direct you to relevant information when you click on them.

# About

Tracktor is an OpenCV based object tracking software. The software is able to perform single-object tracking in noisy environments or multi-object tracking in uniform environments while maintaining individual identities.

Tracktor is command based (i.e. there is no graphical user interface [GUI]) but anyone with basic coding/scripting skills such as [R](https://www.r-project.org/) users can readily use it.

# Requirements

Tracktor works on all operating systems (Windows, Linux and Mac). Below is a step by step guide to install the software and dependencies required to run Tracktor. Most of the code lines you will have to run for the installation must be entered into the Terminal ([Mac](https://www.macworld.co.uk/how-to/mac-software/how-use-terminal-on-mac-3608274/)/[Linux](https://www.howtogeek.com/140679/beginner-geek-how-to-start-using-the-linux-terminal/)) or the [Command Prompt](https://www.digitalcitizen.life/7-ways-launch-command-prompt-windows-7-windows-8) (Windows), which is readily available on your computer. Please note that if you are not familiar with python and/or opencv, this might be a bit overwhelming in the beginning, but we provide enough information to allow you to get started. Once you have overcome the initial challenge to get the software running, you will find that Tracktor is an efficient and versatile tool that can be used for many simple tracking problems. Furthermore, it’s a great way to start learning a bit of coding in python and how object detection algorithms work in general!

# Installation

Follow the steps outlined below. We recommend both installing OpenCV and running Tracktor within a virtual environment using miniconda. A virtual environment can be seen as a closed ‘box’ (i.e the environment) within your computer, in which you can install various things and run software separately from your main system installation. This approach makes it easier to 1) install and 2) prevent potential issues if your machine already runs python (Mac/Linux) or if you have an earlier version of python installed on your machine.

***1) Install miniconda***

Miniconda is an open source distribution of python that aims to simplify package management and deployment. Python is a programming language and requires an interpreter (i.e. software) to run. We recommend using python 3 since the code for Tracktor was designed using this version, but python 2.7+ should also work. Download and install [miniconda with python3](https://conda.io/miniconda.html) from the following link (please note that Anaconda, a more elaborate version of miniconda, will also work in the same manner):

<https://conda.io/miniconda.html>

**\*\*Important:** tick the box "Add Anaconda to my PATH environment variable” when the installer launches, otherwise you will have issues with later stages of this installation tutorial (see step 2 below). The installer states that this is not recommended but the risks to your machine are minimal and establishing a path to Anaconda is needed to run commands from the Terminal ([Mac](https://www.macworld.co.uk/how-to/mac-software/how-use-terminal-on-mac-3608274/)/[Linux](https://www.howtogeek.com/140679/beginner-geek-how-to-start-using-the-linux-terminal/)) or the [Command Prompt](https://www.digitalcitizen.life/7-ways-launch-command-prompt-windows-7-windows-8) (Windows).

***2) create and activate a virtual environment in miniconda***

Creating a virtual environment will allow to install everything you need in this separate “box”, thus preventing any changes to your main system. Once miniconda is installed, you can create a virtual environment with a simple command in the terminal/command prompt.

In the Terminal/Command prompt, type:

conda create --name myenv

(replace “myenv” with any name you want to give this environment)

You will be asked to confirm (proceed ([y]/n)?), type “y”.

That’s it! Your virtual environment has been created. Now you can activate it (=work from within this environment) anytime you want by typing this line in the Terminal/Command prompt:

source activate myenv

(Mac/Linux)

or

activate myenv

(Windows)

(N.B. replace “myenv” with the name of your environment)

**\*\*All the following steps in this tutorial should be performed from within the virtual environment you just created (= after activating it).**

If you want to deactivate the virtual environment, run:

source deactivate (Mac/Linux)

or

deactivate (Windows)

***3) Install the necessary python Packages***

All packages should be installed from within your virtual environment. Activate your virtual environment as explained in section 2.

Once this is done, we will use [Pip](https://pip.pypa.io/en/stable/), a python installer, to install all relevant packages. Run:

conda install pip

If you are having difficulty running Pip, information is available [here](https://pip.pypa.io/en/stable/). To install a package, simply run “pip install packagename” in the Terminal or Command Prompt, from within your virtual environment. Install the following packages (e.g. “pip install numpy”):

* [numpy](http://www.numpy.org/)
* [pandas](https://pandas.pydata.org/)
* [scipy](http://www.scipy.org/)
* [scikit-learn](http://scikit-learn.org/stable/)
* ipython
* ipykernel (“conda install ipykernel” also works if you run into issues with pip)
* [jupyter notebook](http://jupyter.org/install)\*

\*Jupyter notebook is optional. However, it is highly recommended for interactive coding. All example notebooks in this tutorial are written as jupyter notebooks. Jupyter is an application that allows editing and running code in an interactive way, via a web browser (+/- similar to Rcommander for the R users, more information [here](http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html)). For basic information on how to work/run lines or ‘cells’ of code in Jupyter notebook, see [here](http://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/execute.html). For e.g., it is useful to know that you can 1) display line numbers for your code by selecting View > Toggle line numbers from the menu, and 2) use the shortcut keys shift + enter to run a code cell rather than clicking on the button >|Run.

Once all of these packages are installed, create a ‘kernel’ for your virtual environment, which will be necessary for running Tracktor in Jupyter Notebook. Run the following and replace “myenv” with the name you gave your virtual environnement.

python -m ipykernel install --user --name myenv --display-name "python (myenv)”

***4) Install OpenCV***

[OpenCV](https://en.wikipedia.org/wiki/OpenCV) (Open Source Computer Vision) is a library of programming functions focused on real-time computer vision. OpenCV is freeware that works across all platforms but installing it on your computer might be the trickiest part of getting Tracktor to run on your machine. Fortunately miniconda allows installing OpenCV easily. Run:

conda install -c menpo opencv3

or (if this doesn’t work):

conda install -c fonda-forge opencv

If you do not get an error message, OpenCV is now installed in your virtual environment. You can check if the installation worked by typing:

python

import cv2

If these two lines give no error: congratulations, the installation was successful! You can exit Python for now using the exit() command. Note: you entered python by typing ‘python’ above. ‘import cv2’ is used to check if OpenCV works correctly. Once this is done, you need to exit python (with the “exit()” command) to go back to using the normal terminal commands.

It is possible that the installation does not work as described on your machine. If you are having difficulties with this part, please first check online for solutions, or get in touch with an IT specialist at your institution or a computer savvy colleague to help you out. As mentioned, installing OpenCV is the trickiest part of getting Tracktor up and running, but as soon as OpenCV is properly installed, you will be able to get on with your tracking. For more information, see: <https://anaconda.org/menpo/opencv3>

***5) Download Tracktor***

Go on Tracktor’s GitHub page <https://github.com/vivekhsridhar/tracktor> and click the green “clone or download” button. Download the .zip file to your choice location on your computer, and unzip the folder.

Tracktor is basically just a set of lines of code, there is no “installation” required as for most software, and there is no GUI (Graphical User Interface). You will need to enter the various parameters (e.g. name of video, location of the video, etc.) directly into the code, but we will guide you through it for an easy start. For your understanding, here’s a short description of the contents of Tracktor. All the files or folders, which are labelled below as “non-relevant” are not necessary to understand how to work with Tracktor.

[.ipynb\_checkpoints](https://github.com/vivekhsridhar/tracktor/tree/master/.ipynb_checkpoints) non-relevant

[\_\_pycache\_\_](https://github.com/vivekhsridhar/tracktor/tree/master/__pycache__) non-relevant

[logo](https://github.com/vivekhsridhar/tracktor/tree/master/logo) non-relevant

[output](https://github.com/vivekhsridhar/tracktor/tree/master/output) non-relevant

[.DS\_Store](https://github.com/vivekhsridhar/tracktor/blob/master/.DS_Store) non-relevant

[LICENSE](https://github.com/vivekhsridhar/tracktor/blob/master/LICENSE) non-relevant

[README.md](https://github.com/vivekhsridhar/tracktor/blob/master/README.md) This Tutorial

[single\_fish.ipynb](https://github.com/vivekhsridhar/tracktor/blob/master/single_fish.ipynb) Example. Jupyter notebook. This code was created to track a single fish in a noisy environment.

[spider\_track.ipynb](https://github.com/vivekhsridhar/tracktor/blob/master/spider_track.ipynb) Example. Jupyter notebook. This code was created to track 2 spiders of very different sizes, maintaining identities.

[track\_termites.ipynb](https://github.com/vivekhsridhar/tracktor/blob/master/track_termites.ipynb) Example. Jupyter notebook. This code was created to track up to 8 termites of similar sizes, maintaining identities.

[tracktor.ai](https://github.com/vivekhsridhar/tracktor/blob/master/tracktor.ai) non-relevant (Tracktor logo)

[tracktor.pdf](https://github.com/vivekhsridhar/tracktor/blob/master/tracktor.pdf) non-relevant (Tracktor logo)

[tracktor.py](https://github.com/vivekhsridhar/tracktor/blob/master/tracktor.py) This is the actual program. All the functions used in tracktor are defined here. It will not run by itself, but all 3 examples above use the functions available in this file. For unexperienced users, we recommend not touching this file. Instead, **users should pick one of the jupyter notebook examples available above and change the parameters to fit their specific tracking problem.**

# Running Tracktor

***6) Open a jupyter notebook***

We recommend running Tracktor through jupyter notebook. First, ensure that your virtual environment is activated (see section 2). Then, start jupyter notebook by typing:

jupyter notebook

Jupyter notebook’s default directory is the C drive. If you wish to change the directory to a different partition/drive on your computer, instead of typing ‘jupyter notebook’, type:

jupyter notebook --notebook-dir your\desired\directory

(replace ‘your\desired\directory’ with the directory you wish to work in (for e.g. D:\Dropbox))

This will open a window in your browser, displaying your folders. First, we will describe how to run one of the example notebooks provided with Tracktor. Navigate to the Tracktor folder on your machine and open the jupyter notebook titled “single\_fish.ipynb”.

This and other examples are located in the subfolder tracktor-master\examples when you download Tracktor from GitHub in a specified location on your machine.

Jupyter will ask you to select the kernel you want this notebook to run in. It is important that you select your virtual environment because that is where you installed OpenCV and all other packages. If you are not asked directly when the notebook opens, you can change it later by clicking “kernel > change kernel” and selecting your virtual environment in the top menu.

**Important:** you must also specify the file path for Tracktor to locate the example videos or your own videos on your machine. See instructions on how/where to do this in S*ection 7* below.

***7) Running the example code***

The code to run Tracktor is annotated with comments to help your navigate it. You will find explanations for the various functions used, how to change the parameters in these functions, and what each parameter does, directly in the code. If you want further explanations, you can easily access the description of any function in jupyter notebook by placing your cursor inside the parentheses immediately after a function and pressing ‘**shift + tab’** simultaneously. This will display the comments associated with this function from the main code. Please note that the descriptions we provide are only for functions directly associated with Tracktor (all functions starting with “tr.”). Other functions (i.e. from python or OpenCV) can/should be understood by accessing their description as outlined above. Alternatively, you can also find detailed comments in the main code for Tracktor: tracktor.py, which is located in the subfolder tracktor-master\tracktor when you download Tracktor from GitHub.

The Jupyter notebook examples we provide (single\_fish.ipynb, spider\_track.ipynb, track\_termites.ipynb) are organized in 3 sections (called ‘cells’). Reminder: these examples are located in the subfolder tracktor-master\examples.

N.B. In jupyter notebook, clicking in the left part of a cell (white space) selects the cell so that it can be run by pressing the ‘Run’ button in the top bar (the left side of the cell turns blue in this case). Clicking inside a cell (grey space with code) allows editing the code inside the cell to change the tracking parameters, for example (the left side of the cell turns green in this case).

The first cell simply imports the packages/functions necessary to run the Tracktor. If you get an error message after running this cell, it is very likely that one of the packages was not installed properly. If this happens, please go back to sections 3-4 of this tutorial and check that everything has been installed correctly.

The second cell contains the input parameters for tracking. This is where you can choose how many individuals to track, their maximum and minimum sizes (in pixels), the name and file extension of the input video, etc. When using Tracktor with your own videos (not the example videos provided), these are the parameters you will have to tune to fit your problem, by trial and error.

**Important:** Running the tracking examples provided (and, subsequently, your own tracking problems/videos) requires that you change the file path in cell 2 to match the location of the videos on your machine. In the examples provided with Tracktor, the filepath is specified as “ '/home/user/Documents/Vivek/tracktor/videos/' ” when you download the software from Github. This must be changed to the location on your computer where these files have been saved (e.g. 'D:\Dropbox\ Tracktor\tracktor-master\videos' ). Note that the file path must contain forward slashes (/), not backslashes (\). If copying a file path from a Windows OS, change the backslashes to forward slashes. **Also important:** the file path must end with a forward slash (e.g. 'D:/Dropbox/ Tracktor/tracktor-master/videos/'), otherwise the file path won’t be recognized. If Tracktor cannot locate the input video (e.g. because you have improperly specified the file path), you will get an error message saying “Video file cannot be read!”.

The third cell relates to the actual tracking. ***You should not change anything in this cell***, just run it, it will use the parameters you specified in cell 2.

That’s it! You can now run Tracktor on one of the example video of your choice (fish, spider, termites). The tracking parameters (see *Section* 8 below) do not have to be changed for these examples (we have specified them for you); you only need to tell Tracktor what the input and output file paths are in cell 2.

**When Tracktor is running, an additional window opens showing you what is being tracked.** This window might not be immediately obvious and you may have to look for it on your machine (e.g. press Alt+ Tab in Windows to navigate through open windows). Once the tracking is complete, **Tracktor will output 2 files**: one video with the tracking overlaid on it and one .csv file with the XY coordinates from the tracking. Remember that you must specify the file path (in cell 2) where you want Tracktor to save the output files!

***8) Running Tracktor on your own videos***

Once you have familiarized yourself with Tracktor with the examples provided, you can use the code from an example of your choice (e.g. single\_fish.ipynb) and modify the tracking parameters in cell 2 (i.e. block size, offset, min/max\_area) to suit your own purpose. Remember that you must specify the file path (in cell 2) where you want Tracktor to 1) access your videos, and 2) save the output files. Important: if you move files around (e.g. you move the notebook single\_fish.ipynb to a different folder and edit it to fit your own tracking problem), you must also move the program file “tracktor.py” along with it, otherwise the tracking will not work.

In order to track animals in the frame, we convert the original image into a black and white image. This is known as [thresholding](https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html). Post thresholding, we can then classify all black pixels as background and white pixels as the animal. This is why good contrast between background and foreground is essential for Tracktor to work effectively.

To account for slight variation in lighting in the arena, we use a variant of thresholding called [adaptive thresholding](https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html). Adaptive thresholding converts each pixel to black or white based on pixel intensity in its neighbourhood (i.e. surrounding area).

Two parameters in Tracktor control adaptive thresholding:

* ‘**block size**’ determines neighbourhood size
* ‘**offset**’ determines where to set the threshold relative to the neighbourhood mean

Clicking the hyperlink above will provide a more detailed description of these parameters.

Post thresholding, we may have more than just the animal as white pixels. But since we know how big our animals are, we can create a size cutoff above and below which white pixel blobs (which might not be the animal) are ignored.

Two parameters in Tracktor determine this size cutoff:

* ‘**min\_area**’, which is the minimum area (in pixels) that the animal might take up
* ‘**max\_area**’, which is the maximum area (in pixels) that the animal might take up

Play around with these four tracking parameters in cell 2 to identify which values produce the best tracking results for your particular situation.

**Note**: You’re bound to run into errors and failed tracking as you try to find the right parameter values for your videos. Whenever the tracking fails, or if it’s running but you’re not satisfied with the output, restart the kernel (in the top bar in jupyter). Restarting the kernel kills all the running processes. You can then change the tracking parameters and start fresh for a new try. Note that you’ll have to re-run the first cell after restarting the kernel.  Sometimes, Tracktor will succeed at tracking the animal(s) for part of the video and then crash. This means that, at some point during the video, the tracking parameters are no longer adequate (e.g. the lighting/contrast changed). Continue modifying the parameters until you can successfully track the animal(s) throughout the entire video and you are satisfied with the output.

Tracktor is very likely to start tracking stuff you do not want it to (e.g. reflection on the water surface). Try to get rid of these with the right size thresholds (i.e. change the values for ‘min\_area’ and ‘max\_area’).

Finding the right combination of parameter values (i.e. block size, offset, min/max\_area) for your tracking problem is a trial-and-error process. After a few trials, you will get a good feeling for what values should be used to optimize the tracking for your specific videos. It is difficult for us to provide a range of values that the three parameters must take for your specific tracking problem(s) because that will depend on factors such as the resolution of your videos (e.g. SD vs. HD vs. 4k), the number of pixels representing the animal to be tracked, the lighting conditions and contrast in the video, etc. Again, determining the right set of parameter values is a trial-and-error process that is usually solved relatively quickly and can then be applied across similar videos.

**Summary statistics:** Based on the xy coordinates (in pixels) outputted by Tracktor, summary statistics (e.g. cumulative distance traveled, maximum speed, time spent in a given area) can be computed for the examples provided and for your own videos. These stats are first provided in units of pixels and frames. To convert these measurements into length (centimeters) and time (seconds) units, specify the ‘framerate’ and ‘pxpercm’ parameters below the cell that states “Fill in the parameters below if you'd like movement measures to be converted from pixels and frames to real-world measures […]”.

The ‘**framerate**’ parameter (in frames per second; fps) was set by you in the camera settings at the time of filming (e.g. 30, 120, 500, 1000 fps).

The ‘**pxpercm**’ parameter corresponds to the scale (i.e. the number of pixels in a given unit of length). To determine this scale, open your video in a software that allows you to measure an object of a known length (e.g. 5 cm) and tell you how many pixels corresponds to this length. Freeware that allows you to do this easily is [ImageJ](https://imagej.nih.gov/ij/). After you have opened a video in imageJ, use the line tool to measure your object of known dimensions, then click on Analyze > Set Scale. This will indicate a ‘Distance in pixels’.

**Note:** ImageJ can only open import .avi files with specific codecs (e.g. MJPEG). If you cannot open you video, convert it into an image stack (individual frames) or a .avi file with the codec MJPEG. This can be done easily with the freeware [FFmpeg](https://www.ffmpeg.org/) (or any other software of your choice). FFmpeg is a powerful, versatile solution to convert video – it is code-based but there are several GUI available if you prefer this option. An excellent one for Windows is [myFFmpeg](http://www.myffmpeg.com/). Others are available; simply search FFmpeg GUI in Google.

**9. *Common problems and fixes***

If you encounter issues or cryptic error messages we recommend searching in Google for answers. There is a large community of people using OpenCV-based solutions for tracking and most issues have a solution on Stack Overflow. Additionally, below are some examples of issues you might encounter and fixes. If your problems persist, don’t hesitate to get in touch with us by [email](mailto:vsridhar@orn.mpg.de) or by reporting an issue on [GitHub](https://github.com/vivekhsridhar/tracktor). We will do our best to help you get started with Tracktor.

*I get one or several error message(s) after running cell 1 for one of the examples provided with Tracktor.*

-> One of the packages/dependencies was probably not installed properly. Return to   
*Sections 3-4* of this tutorial and verify that all packages were installed correctly.

*I get the error message* “Video file cannot be read!” *after running cell 3 for one of the examples provides with Tracktor (or my own tracking problem).*

-> This mean that Tracktor cannot locate the input file/video. Make sure that the file path to the video (i.e. where the video is located on your machine) is correctly specified in cell 2. See *section 7* of the tutorial, above. For example, change the file path from '/home/user/Documents/Vivek/tracktor/videos/' (the one specified when you download the software from Github) to the location on your computer where these files have been saved (e.g. 'D:\Dropbox\ Tracktor\tracktor-master\videos’). Note that the file path must contain forward slashes (/), not backslahes (\). If copying a file path from a Windows OS, change the backslahes to forward slashes. **Important:** the file path must also end with a forward slash (e.g. 'D:/Dropbox/ Tracktor/tracktor-master/videos/’)

*I get one or several error messages after running cell 3 for my own tracking problem(s).*

-> The tracking parameters you specified are probably incorrect. For instance, if there are too many targets detected, or no targets detected, this can cause Tracktor to crash. The solution is to try other parameter values. For that, restart the kernel, input other parameters, and re-run the code (including cell 1) until you get the desired output.

*Tracktor crashes after tracking the animal(s) successfully over some period of time.*

-> Sometimes, Tracktor will succeed at tracking the animal(s) for part of the video and then crash. This is a very good start but it means that, at some point during the video, the tracking parameters are no longer adequate (e.g. the lighting/contrast changed). Continue modifying the parameters until you can successfully track the animal(s) throughout the entire video and you are satisfied with the output.

*I cannot run python through the Git Bash terminal.*

-> This seems to be a problem with Git Bash. Run python from the Mac/Linux Terminal or Windows Command Prompt and that will solve it.