TRACKTOR v0.1 - tutorial

Last updated: 03.2018

If you find errors in this tutorial and/or identify links that are no longer working or outdated information, please notify Vivek Sridhar by [email](mailto:vsridhar@orn.mpg.de) or by reporting an issue on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.

# 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 not 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 everything required to run Tracktor. Please note that most of the lines of code you will have to run for installation have to be run in 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 in your computer.

# Installation

Follow the steps outlined below. We recommend to install opencv and run Tracktor within a virtual environment using miniconda. This makes it 1) easier to install and 2) potentially prevents many troubles if your system already uses python (Mac/Linux) or that you already have another python version installed.

*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. I recommend using Python 3 since the code for Tracktor was designed in this version, but Python 2.7+ should also work. Download and install miniconda with Python3 from here (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 that says "Add Python X.X to PATH” when the installer launches otherwise you will have issues running pip (see step 2 below).

*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 to make any changes in your main system. Once miniconda is installed, you can create a virtual environment with the command in the terminal/command prompt:

conda create --name myenv

(replace “myenv” with any name you want to give to 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 with the command:

source activate myenv (Mac/Linux)

or

activate myenv (Windows)

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

**All the following steps in this tutorial should be performed from within the virtual environment (= 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*

[Pip](https://pip.pypa.io/en/stable/) can be used to install all packages. Pip is typically already installed with Python, but if you are having difficulty running it, 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. 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/)
* [Jupyter notebook](http://jupyter.org/install)\*

\*Jupyter notebook is optional. However, it is highly recommended it for interactive coding. All example notebooks are written as jupyter notebooks, and it also makes it generally easier to run code.

*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 has an easy-to-use installation of OpenCV. Run:

conda install -c menpo opencv3

If you get no 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.

It is possible that the installation does not work on your machine as described. If you struggle with this part, please check online for solutions, or get in touch with a IT specialist or a computer savvy colleague to help you out. As mentioned, this is the trickiest part of getting Tracktor up and running, but as soon as OpenCV is properly installed, you will be able to track videos. More information: <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 I will guide you through it for an easy start. For your understanding, here’s a short description of the content of Tracktor. All the files or folders labelled here 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 2 spiders of very different 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, I recommend not touching this file, but rather pick one of the jupyter notebooks available above and change the parameters to fit your actual tracking problem.

# Running Tracktor

*6) Open a Jupyter notebook*

We recommend running Tracktor through Jupyter. For this, first make sure your virtual environment is activated (see section 2). Then start Jupyter by typing:

jupyter notebook

This will open a window in your browser, displaying your folders. We will first describe here how to run one of the example notebooks we provide in tracktor. Navigate to the tracktor folder and open the jupyter notbook entitled “single\_fish.ipynb”. Jupyter will ask you to select the kernel you want this notebook to run in. It’s important that you select your virtual environment because that is where your installation of OpenCV is. If you are not asked directly when the notebook opens, you can change it later by clicking on the top menu on “kernel > change kernel” and selecting your virtual environment.

*6) Running the example code*

The code is fully functional, but we can (and often have to), tweak it a little bit. An important thing to know about the tracking solution we use here is that it will track the first object it detects throughout the entire video, and not care about the rest (more or less). So it’s important that the beginning of the video is a static as possible (no water ripples etc.), otherwise it might start tracking something else than the fish. I’m not going to explain it all here, but here are a few important lines in the code for our use:

*import numpy as np*

*import pandas as pd*

*import cv2*

*from scipy.optimize import linear\_sum\_assignment*

*from scipy.spatial.distance import cdist*

This first section is just to load the modules we’ll be using later, just as loading libraries in R. If you run into an error saying that a module is missing, just type “pip3 install module\_name” and that should sort it.

*cap = cv2.VideoCapture("/Users/simongingins/Dropbox/Postdoc/FS-gobies/Test1.mp4")*

Obviously in here you should replace the path with the path of the video you want to analyse.

*thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY\_INV, 31, 38)*

This is THE important line for us. This codes for the adaptive threshold, which is the technique we’re using to detect the fish. The last two numbers in the parentheses (here 31 & 38) are the two parameters we have to play with to get the best output. The second one (38 here) is the actual threshold. Basically, if you lower the value, it will detect smaller contrasts, and vice versa. So, if you have either too many things detected, or that only parts of your fish that are detected, this is the value to play with. The first value (here 31) determines the size, in pixels, of small boxes in which adaptive thresholding is done. I don’t want to get into the details here (check online what adaptive thresholding is if you’re interested). But basically if you have varying light conditions in your video, such as some shaded areas, this is a parameter that might increase the performance.

In summary, play with these two values until you are happy with the result!

*#'''*

This tiny line is present twice in the current code, once before the section “Apply the hungarian algorithm to maintain identities” and once after the section “Draw centroids”. Just as in R, a # means that the line is not taken into account in the code (used mostly for annotating the code). Now the *'''* also means the same, but for an entire section.

What this means is that, in its current state, the full code will run and try to track the centroid of your fish at the same time. This can be problematic if you haven’t figured out the right parameters for the thresholding yet, as whenever the fish “disappears” the program will crash.

So a good way to get around this is to take out the tracking part of the code when you’re trying to find the right threshold parameters, and only run the full code when you are happy with the thresholding. To do this, simply delete the # in each of the two #''' lines (just leaving the '''), and the code will skip this section.

*cv2.imshow('frame', final)*

Here, you can change “final” to one of the following: “blur”, “thresh”, “grey”, “final”.

Final: will display your video, overlaid with the contours of the detected areas, and the centroid of the fish.

Blur: will display the blurring process, which is a step we do prior to thresholding (not very relevant to display I guess).

Thresh: will display the output of the thresholding, in black & white. This is really handy when you’re trying to find the right parameters.

Grey: will display the image in greyscale, which is also a step we do prior to thresholding. Same as for blurring, probably not very relevant to display.

*#fourcc = cv2.VideoWriter\_fourcc(\*'avc1')*

*#out = cv2.VideoWriter(filename = 'fish.mp4', fourcc = fourcc, fps = 30.0, frameSize = (960, 540), isColor = True)*

These two lines (currently not implemented, because of the #), will create a video output and save it in the same folder as the original video. Here it is named “fish.mp4” but you can give it any name. If you want to save this output, simply delete the # in the beginning of these two lines, AND in the beginning of the following two lines at the end of the code:

*#out.write(final)*

*#out.release()*

The software will also create a .csv file with the x,y positions of the centroid, saved in the local folder.