STracking: a free and open-source Python library for particle tracking and analysis

Sylvain Prigent, Ludovic Leconte, Cesar Augusto Valades-Cruz, Charles Kervrann, Jean Salamero

STracking: a free and open-source Python library for particle tracking and analysis

Sylvain Prigent[1,2], Cesar Augusto Valades-Cruz[1,2], Ludovic Leconte[1,2], Jean Salamero[1,2] and Charles Kervrann [1,2]

[1]SERPICO Project Team, Inria Centre Rennes-Bretagne Atlantique, F-35042 Rennes, France and [2]SERPICO Project Team, UMR144 CNRS Institut Curie, PSL Research University, F-75005 Paris, France

Abstract

Summary: Analysis of intra- and extracellular dynamic like vesicles transport involves particle tracking algorithms. The design of a particle tracking pipeline is a routine but tedious task. Therefore, particle dynamics analysis is often performed by combining several pieces of software (filtering, detection, tracking, etc.) requiring many manual operations, and thus leading to poorly reproducible results. Given the new segmentation tools based on deep learning, modularity and interoperability between software have become essential in particle tracking algorithms. A good synergy between a particle detector and a tracker is of paramount importance. In addition, a user-friendly interface to control the quality of estimated trajectories is necessary. To address these issues, we developed STracking, a Python library that allows combining algorithms into standardized particle tracking pipelines.

Availability and implementation: STracking is available as a Python library using 'pip install' and the source code is publicly available on GitHub (https://github.com/sylvainprigent/stracking). A graphical interface is available using two napari plugins: napari-stracking and napari-tracks-reader. These napari plugins can be installed via the napari plugins menu or using 'pip install'. The napari plugin source codes are available on GitHub (https://github.com/sylvainprigent/napari-tracks-reader, https://github.com/sylvainprigent/napari-stracking).

Contact: sylvain.prigent@inria.fr or cesar-augusto.valades-cruz@curie.fr

Supplementary information: Supplementary data are available online.

Introduction

The study of cell biology dynamics, such as intracellular membrane transport inward (i.e. endocytosis) and outward (i.e. exocytosis/recycling), has been difficult or at least incomplete until recently, due to the heterogeneity of the motion behavior of the analyzed structures. Different tracking software have been published [e.g. u-track (Jaqaman *et al.*, 2008), TrackMate (Tinevez *et al.*, 2017), MTT (Sergé *et al.*, 2008), MSSEF-TSAKF (Jaiswal *et al.*, 2015), maptrack (Feng *et al.*, 2011), Cell Tracking Profiler (Mitchell *et al.*, 2020), btrack (Ulicna *et al.*, 2021)] to track individual biomolecules or extended objects with a shape, such as cells [e.g. TrackMate (Ershov *et al.*, 2021) and CellProfiler (Stirling *et al.*, 2021)], to obtain spatial information and to quantify their kinetics. Most of them focus on the accuracy and reproducibility of the analysis but the user interfaces remain complex or even limited to a two-dimensional representation. The development of a user-friendly graphical user interface (GUI) therefore appears necessary to facilitate the selection of parameters, the analysis and the visualization of 3D + time trajectories estimated from complex 3D videos. The use of Python, a versatile and free programming language is growing rapidly within the bioimaging user community (Fernandez-Gonzalez *et al.*, 2022). Python tools for visualization [e.g. napari (Sofroniew *et al.*, 2021), ipyvolume (Breddels *et al.*, 2018), SeeVis (Hattab & Nattkemper, 2019)] and analysis [ZeroCostDL4Mic (von Chamier *et al.*, 2021), BioImageIT (Prigent *et al.*, 2021), Cellpose (Stringer *et al.*, 2021)] are widely applied to microscopy images.

On the other hand, a lot of particle tracking approaches have been developed over the last decades. Interestingly, although a number of studies aimed at comparing particle tracking performance have been published (Carter *et al.*, 2005; Cheezum *et al.*, 2001; Chenouard *et al.*, 2014; Ruusuvuori *et al.*, 2010; Smal & Meijering, 2015; Smal *et al.*, 2010), none of the tested methods seems to perform in a generic way, regardless of the type of image data. As a consequence, it is critical for users to have the possibility to test different detectors and/or trackers in order to identify the best solution for their application. In addition, efforts in the Python community includes particle tracking packages such as TrackPy (Allan *et al.*, 2021). TrackPy is a complete particle tracking toolkit, but the code can be a barrier for non-expert user.

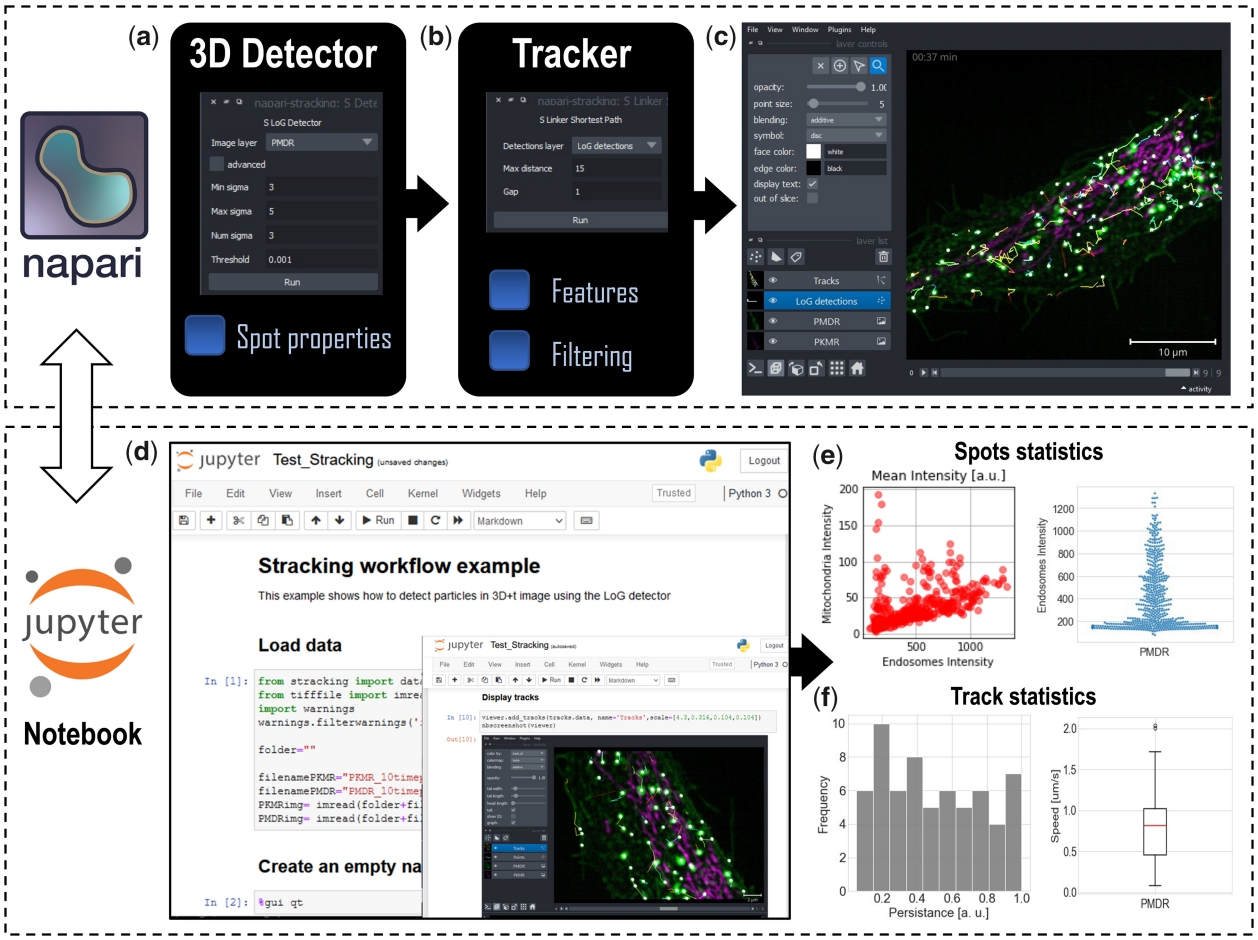

Fig. 1. Overview of STracking library implemented in its napari plugin (**a–c**) and Jupyter Notebook (Kluyver *et al.*, 2016) (**d–f**). Fifty-five planes 3D volumes of live RPE1 cells double stained with PKMR for Mitochondria (magenta) and with plasma membrane deep red (PMDR) for endosomal pathway (green) were acquired within 4.3 s per stack using Lattice light-sheet Structure Illumination Microscopy (LLS-SIM). STracking workflow is illustrated here with single particle tracking of endosomal pathway (PMDR). napari-stracking includes spots detection (a) and linking (b) through a GUI. 3D data and tracks are rendered using napari viewer (c). Jupyter notebook (d) allows also spots (e) and tracks (f) analysis. Additionally, they permit to get spots properties and tracks features, as well as tracks filtering. LLS-SIM data were reconstructed using MAP-SIM (Křížek *et al.*, 2016) (A color version of this figure appears in the online version of this article.)

Here, we present STracking, an open-source Python library for combining algorithms into standardized particles tracking pipelines for biological microscopy images. STracking is distributed under a BSD 3-Clause 'New' or 'Revised' License. STracking combines particle detection, tracking and analysis methods and can be used via a napari plugin. STracking contributes to the recent ecosystem of Python-based plugins for bioimage analysis.

Implementation and application

STracking breaks down a particle tracking pipeline into five components: (i) frame-by-frame particle detection; (ii) particle linking; (iii) analysis of particle properties; (iv) design of track features; and (v) filtering of tracks. Each component is represented as a Python object.

Each component can be implemented separately. This modular design makes it easy to update and facilitates interoperability with other plugins/algorithms. Whenever a new detection or tracker algorithm is added, compatibility is guaranteed with the particle tracking pipeline, and it is versioned within the STracking library. Several particle detectors are available in STracking: Difference of Gaussian, Determinant of Hessian and Laplacian of Gaussian from the Python library scikit-image (van der Walt *et al.*, 2014). In addition, STracking includes a mask detector, called SSegDetector. This detector takes a label or binary mask as input image and returns a list of object positions (centroids of connected components). Moreover, STracking includes a tracker (Matov *et al.*, 2011) that estimates the optimal tracks as follows: first, a connection graph is created with all the possible connections. Second, tracks are iteratively extracted from the graph using shortest path and graph pruning.

STracking library uses two data structures: 'SParticles' to manage the set of detected particles and 'STracks' to manage the collection of trajectories. These data structures contain SciPy (Virtanen *et al.*, 2020) objects to store the particles and the tracks. The particles are represented with a 2D numpy array where each row is dedicated to a specific particle and columns are [T, Z, Y, X] for 3D data and [T, Y, X] for 2D data. The properties of particles are stored in a dictionary. Similarly, tracks are stored in a 2D numpy array where each row is dedicated to specific particle and columns are [trackID, T, Z, Y, X] for 3D data and [trackID, T, Y, X] for 2D data. Tracks features and split/merge events are stored using dictionaries. This data representation is the same as napari (Sofroniew *et al.*, 2021) points and tracks layers, making STracking natively compatible with the napari viewer. We thus implemented a STracking napari plugin suite (napari-tracks-reader, napari-stracking). It provides a graphical interface to create a STracking pipeline without writing Python code. STracking could be used as script in Python or napari plugin. The STracking library can be combined with other Python packages to extend STracking functionalities. The napari plugin allows one to perform a full STracking pipeline, or to load detections or tracks from another software such

as StarDist (Schmidt *et al.*, 2018), TrackMate (Tinevez *et al.*, 2017) or u-track (Jaqaman *et al.*, 2008) and continue the analysis with STracking and napari. Documentation on STracking library with examples is available at https://sylvainprigent.github.io/stracking/. STracking documentation was created using sphinx and the autodoc extension.

STracking pipeline using the napari plugin is illustrated with data obtained in Lattice Light-Sheet Structured Illumination Microscopy (Chen *et al.*, 2014) (Fig. 1 and Supplementary Video S1). The STracking workflow could also be implemented using Jupyter notebook (Supplementary Note S1). Additionally, STracking library can be used for cell migration experiments (Supplementary Note S2) using label mask images produced by other software such as CellPose (Stringer *et al.*, 2021) or StarDist (Schmidt *et al.*, 2018) through napari-stracking plugin, Jupyter Notebook or Python scripting. These examples demonstrate the ability of STracking to analyze complex datasets acquired with most advanced microscopy technologies.

Conclusions

The STracking library simplifies the design of single particle tracking workflows through a graphical interface using napari and a comprehensive Python library of functions. Unlike previous single particle tracking tools in Python ecosystem, it provides a very flexible solution for processing and visualizing the tracks taking advantage of Napari (Sofroniew *et al.*, 2021) viewer for 3D + time representation. A similar approach was introduced in TrackMate software (Tinevez *et al.*, 2017) for the visualization and validation of 2D tracks in Fiji (Schindelin *et al.*, 2012) java-based environment. Thus, reproducible analysis can be performed without being an expert programmer. For this purpose, STracking library includes a pipeline class to allow executing a tracking pipeline recorded as a json file. We would point out that this plugin-implemented recording technique is not an optimal software architecture since it should be done by the host platform. To overcome this difficulty, we recommend using a powerful data management software such as the recent BioImageIT platform (Prigent *et al.*, 2021).

In summary, the STracking library greatly simplifies the inspection and optimization of single particle tracking algorithms and thus allows the evaluation of new detection and tracker algorithms in this context, which are constantly being developed.

Funding

This work was supported by the French National Research Agency (France-BioImaging Infrastructure [ANR-10-INBS-04-07] and LabEx Cell(n)Scale [ANR-11-LABX-0038] as part of the IDEX PSL [ANR-10-IDEX-0001-02]).

Data availability

The data underlying this article are available in FigShare, at https://doi.org/10.6084/m9.figshare.19322171.

References

Allan,D.B. *et al.* (2021) Trackpy v0.5.0. https://doi.org/10.5281/ZENODO.4682814 (11 March 2022, date last accessed).

Breddels,M. *et al.* (2018) ipyvolume v0.4.5. https://doi.org/10.5281/ZENODO.1286976 (11 March 2022, date last accessed).

Carter,B.C. *et al.* (2005) Tracking single particles: a user-friendly quantitative evaluation. *Phys. Biol.*, **2**, 60–72.

Cheezum,M.K. *et al.* (2001) Quantitative comparison of algorithms for tracking single fluorescent particles. *Biophys. J.*, **81**, 2378–2388.

Chen,B.-C. *et al.* (2014) Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution. *Science*, **346**, 1257998.

Chenouard,N. *et al.* (2014) Objective comparison of particle tracking methods. *Nat. Methods*, **11**, 281–289.

Ershov,D. *et al.* (2021) Bringing TrackMate into the era of machine-learning and deep-learning. *bioRxiv*, 2021.09.03.458852.

Feng,L. *et al.* (2011) Multiple dense particle tracking in fluorescence microscopy images based on multidimensional assignment. *J. Struct. Biol.*, **173**, 219–228.

Fernandez-Gonzalez,R. *et al.* (2022) PyJAMAS: open-source, multimodal segmentation and analysis of microscopy images. *Bioinformatics*, **38**, 594–596.

Hattab,G. and Nattkemper,T.W. (2019) SeeVis—3D space-time cube rendering for visualization of microfluidics image data. *Bioinformatics*, **35**, 1802–1804.

Jaiswal,A. *et al.* (2015) Tracking virus particles in fluorescence microscopy images using multi-scale detection and multi-frame association. *IEEE Trans. Image Process.*, **24**, 4122–4136.

Jaqaman,K. *et al.* (2008) Robust single-particle tracking in live-cell time-lapse sequences. *Nat. Methods*, **5**, 695–702.

Kluyver,T. *et al.* (2016) Jupyter notebooks—a publishing format for reproducible computational workflows. In: Loizides, F. and Schmidt, B. (eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Göttingen, Germany. IOS Press, Amsterdam, Netherlands, pp. 87–90.

Křížek,P. *et al.* (2016) SIMToolbox: a MATLAB toolbox for structured illumination fluorescence microscopy. *Bioinformatics*, **32**, 318–320.

Matov,A. *et al.* (2011) Optimal-flow minimum-cost correspondence assignment in particle flow tracking. *Comput. Vis. Image Underst.*, **115**, 531–540.

Mitchell,C. *et al.* (2020) Cell tracking profiler—a user-driven analysis framework for evaluating 4D live-cell imaging data. *J. Cell Sci.*, **133**, jcs241422.

Prigent,S. *et al.* (2021) BioImageIT: Open-source framework for integration of image data-management with analysis. *bioRxiv*, 2021.12.09.471919.

Ruusuvuori,P. *et al.* (2010) Evaluation of methods for detection of fluorescence labeled subcellular objects in microscope images. *BMC Bioinformatics*, **11**, 248.

Schindelin,J. *et al.* (2012) Fiji: an open-source platform for biological-image analysis. *Nat. Methods*, **9**, 676–682.

Schmidt,U. *et al.* (2018) Cell detection with star-convex polygons. In: Frangi,A.F. *et al.* (Eds.), *Medical Image Computing and Computer Assisted Intervention—MICCAI 2018, Granada, Spain*. Springer International Publishing, Cham, Switzerland, pp. 265–273.

Sergé,A. *et al.* (2008) Dynamic multiple-target tracing to probe spatiotemporal cartography of cell membranes. *Nat. Methods*, **5**, 687–694.

Smal,I. and Meijering,E. (2015) Quantitative comparison of multiframe data association techniques for particle tracking in time-lapse fluorescence microscopy. *Med. Image Anal.*, **24**, 163–189.

Smal,I. *et al.* (2010) Quantitative comparison of spot detection methods in fluorescence microscopy. *IEEE Trans. Med. Imaging*, **29**, 282–301.

Sofroniew,N. *et al.* (2021) napari: 0.4.12rc2. https://doi.org/10.5281/ZENODO.5587893 (11 March 2022, date last accessed).

Stirling,D.R. *et al.* (2021) CellProfiler 4: improvements in speed, utility and usability. *BMC Bioinformatics*, **22**, 433.

Stringer,C. *et al.* (2021) Cellpose: a generalist algorithm for cellular segmentation. *Nat. Methods*, **18**, 100–106.

Tinevez,J.-Y. *et al.* (2017) TrackMate: an open and extensible platform for single-particle tracking. *Methods*, **115**, 80–90.

Ulicna,K. *et al.* (2021) Automated deep lineage tree analysis using a Bayesian single cell tracking approach. *Front. Comput. Sci.*, **3**, 734559.

van der Walt,S. *et al.* (2014) Scikit-image: image processing in Python. *PeerJ*, **2**, e453.

Virtanen,P. *et al.* (2020) SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods*, **17**, 261–272.

von Chamier,L. *et al.* (2021) Democratising deep learning for microscopy with ZeroCostDL4Mic. *Nat. Commun.*, **12**, 2276.