SOFTWARE TOOL ARTICLE

REVISED TRAIT2D: a Software for Quantitative Analysis of Single Particle Diffusion Data [version 2; peer review: 2 approved]

Francesco Reina[1], John M.A. Wigg[2], Mariia Dmitrieva[3], Bela Vogler[2], Joël Lefebvre[4], Jens Rittscher[3], Christian Eggeling[1,2,5]

[1]Leibniz-Institut für Photonische Technologien e.V, Jena, Germany
[2]Institute of Applied Optics and Biophysics, Friedrich-Schiller-Universität, Jena, Germany
[3]Department of Engineering Science, University of Oxford, Oxford, UK
[4]Département d'informatique, University of Quebec at Montreal, Montreal, Canada
[5]MRC Human Immunology Unit, Weatherall Institute of Molecular Medicine, University of Oxford, Oxford, UK

v2 **First published:** 20 Aug 2021, **10**:838
https://doi.org/10.12688/f1000research.54788.1
Latest published: 31 Jan 2022, **10**:838
https://doi.org/10.12688/f1000research.54788.2

Abstract

Single particle tracking (SPT) is one of the most widely used tools in optical microscopy to evaluate particle mobility in a variety of situations, including cellular and model membrane dynamics. Recent technological developments, such as Interferometric Scattering microscopy, have allowed recording of long, uninterrupted single particle trajectories at kilohertz framerates. The resulting data, where particles are continuously detected and do not displace much between observations, thereby do not require complex linking algorithms. Moreover, while these measurements offer more details into the short-term diffusion behaviour of the tracked particles, they are also subject to the influence of localisation uncertainties, which are often underestimated by conventional analysis pipelines. we thus developed a Python library, under the name of TRAIT2D (Tracking Analysis Toolbox – 2D version), in order to track particle diffusion at high sampling rates, and analyse the resulting trajectories with an innovative approach. The data analysis pipeline introduced is more localisation-uncertainty aware, and also selects the most appropriate diffusion model for the data provided on a statistical basis. A trajectory simulation platform also allows the user to handily generate trajectories and even synthetic time-lapses to test alternative tracking algorithms and data analysis approaches. A high degree of customisation for the analysis pipeline, for example with the introduction of different diffusion modes, is possible from the source code. Finally, the presence of graphical user interfaces lowers the access barrier for users with little to no programming experience.

Keywords

Single Particle Tracking, Diffusion, Data analysis, Python, Graphical User Interface, Simulation, Microscopy

Open Peer Review

Approval Status ✓ ✓

	1	2
version 2 (revision) 31 Jan 2022	✓ view	
version 1 20 Aug 2021	? view	✓ view

1. **Xavier Michalet**, University of California, Los Angeles (UCLA), Los Angeles, USA
University of California at Los Angeles, Los Angeles, USA

2. **Eva Christensen Arnspang**, University of Southern Denmark, Odense, Denmark

Any reports and responses or comments on the article can be found at the end of the article.

This article is included in the Python collection.

Corresponding authors: Francesco Reina (francesco.reina@ipht-jena.de), Christian Eggeling (Christian.Eggeling@leibniz-ipht.de)

Author roles: Reina F: Conceptualization, Methodology, Project Administration, Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; **Wigg JMA**: Conceptualization, Methodology, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Dmitrieva M**: Conceptualization, Methodology, Software, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Vogler B**: Methodology, Software; **Lefebvre J**: Conceptualization, Methodology, Software, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Rittscher J**: Funding Acquisition, Project Administration, Supervision, Writing – Review & Editing; **Eggeling C**: Funding Acquisition, Project Administration, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: FR, CE and JMAW were supported by Wolfson Foundation, the Medical Research Council (MRC, grant number MC \textunderscore UU \textunderscore 12010/unit programmes G0902418 and MC \textunderscore UU \textunderscore 12025), MRC/BBSRC/EPSRC (grant number MR/K01577X/1), the Wellcome Trust (grant ref. 104924/14/Z/14), the Deutsche Forschungsgemeinschaft (Research unit 1905 ``Structure and function of the peroxisomal translocon'', Jena Excellence Cluster ``Balance of the Microverse'', Collaborative Research Center 1278 ``Polytarget''), Oxford-internal funds (John Fell Fund and EPA Cephalosporin Fund) and Wellcome Institutional Strategic Support Fund (ISSF). JL was supported by a FRQNT postdoctoral fellowship (257844). MD and JR were funded by a Wellcome Collaborative award (203285)
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Reina F, Wigg JMA, Dmitrieva M *et al.* **TRAIT2D: a Software for Quantitative Analysis of Single Particle Diffusion Data [version 2; peer review: 2 approved]** F1000Research 2022, **10**:838 https://doi.org/10.12688/f1000research.54788.2

First published: 20 Aug 2021, **10**:838 https://doi.org/10.12688/f1000research.54788.1

REVISED **Amendments from Version 1**

We have refined the manuscript to include additional details on the theoretical foundation of the work and words of caution for inexperienced users, following the peer review reports. In particular, we state clearly, following reviewer's feedback, that in analyzing Brownian diffusing particles' motion, software users should take care in using an appropriate number of data points. More information about this topic can be found in the expanded list of references, which will guide them toward a more detailed understanding of the theory underlying Single Particle Tracking and the analysis of experimental data collected thereof. We further elaborated on the motivation of the choices made in the production of the code, to make clarity on the driving ideas behind the design of TRAIT2D. The underlying code has been also modified to improve more functionalities in the GUI, correct bugs that were affecting it, and provide expanded functionality on the tracker side. At the release of this manuscript, no additional tracking algorithms are provided, but the authors are working on additional functionalities to add new features to the code. More details on the changes are present in the responses to the referee reports, which we invite the reader to consult.

An updated version of the software (1.2) has been submitted concurrently with the release of this version following the submission of this version of the paper, which includes many of the changes suggested by the reviewers. Additional functionalities that have been suggested have not been implemented at present, but are nevertheless under study and should be included in future releases. A considerable effort has been put by the additional author, B. Vogler, on code review and on improvement of the analysis software performance to improve the user experience.

Any further responses from the reviewers can be found at the end of the article

Introduction

Single particle tracking (SPT) is one of the most direct and employed methods to quantify particle dynamics in a sample using optical microscopy. As the name suggests, this approach relies on the identification and tracking of single particles in a sample, followed by the analysis of the detected trajectories. The analysis of particle trajectories is usually carried out using the description given by Einstein and Smoluchowski of Brownian motion[1,2] to derive the diffusion coefficient of the objects. Given its nature as a data analysis method rather than a technique onto itself, SPT is indifferent to how the particle is detected, as long as the signal-to-noise levels are sufficient to identify it against the background. As a consequence of this, it is almost always necessary to label the sample with adequate target specificity, and it is important that the labelling should be sparse enough to enable single molecule level of detail, particularly when employing conventional, diffraction-limited microscopy.[3,4]

Among the earliest example of SPT experiments, we can mention the testing of Einstein's theory of Brownian motion,[5] during which tracking was executed by purely analog means,*i.e.* using pen and paper. The introduction of electronic digital camera detection and innovative microscopy systems have made it possible to extend the range of applications of SPT, which today reaches framerates up to 50kHz.[6] More importantly, the use of large scattering tags is nowadays not necessary anymore to reach nanometer levels of localisation precision.[7,8] The rise of such SPT-capable techniques, characterized by fast sampling and high data throughput, thus generates the need to adapt previously developed tracking and analysis pipelines.

Conventionally, trajectory analysis in diffusive systems is performed by monitoring the Mean Squared Displacement of the particle against the time increments, and analyzing them through the Brownian motion[1] and similar models of diffusion.[9,10] While this kind of pipeline has the advantage of being straightforward and easy to implement, it is not immune from potential pitfalls. As we elaborate further in the Methods section, the localisation error is a critical factor that is often overlooked.[11] This aspect has become especially critical in recent times, given the rise in sampling rates accessible with modern hardware. Motion blurring, which is inherent in SPT given the movement of the target particles, is also often neglected as a potential source of measurement error,[12] which can also lead to disruptive effects in the estimation precision of relevant physical parameters. Overlooking these sources of error leads to overestimation of the detected diffusion coefficients,[13] or to erroneously detected subdiffusion.[14] These spurious deviation arise particularly at short time ranges, given that the particle displacements at these time frames is comparable to the localisation noise, and is thus particularly dangerous since many analysis pipelines often rely on the very first data points to extract relevant diffusion parameters.[15] In addition to this, another factor that affects the correct estimation of physical quantities is the model fitting methodology employed and the number of Mean Squared Displacement points to employ in the data analysis pipeline, which require careful consideration on an ad-hoc basis for each dataset.[16–18]

Numerous toolboxes and algorithms have been produced to fulfill the need to identify and track single particles,[19–21] and analyse their motion employing conventional mean squared displacement analysis,[22]machine and deep learning[23–25] and other methods.[26,27] Oftentimes the source code is shared by the authors and is readily available, and while in some cases they are not platform-agnostic and require licensed software to be executed, on the other hand, e.g., web-based tools and open source alternatives have also been developed to widen the access possibility to such tools. On the other hand,

tools available on open-source platforms, such as FIJI[28] or ICY,[29] are often more focused on the particle tracking, rather than the data analysis itself.

The TRAIT2D (Tracking Analysis Toolkit - 2D version), hereby presented, aims to supply the scientific community with an accessible, open-source, and platform-agnostic tool for particle tracking, simulation and analysis in two dimensions. In TRAIT2D, intuitive graphical user interfaces facilitate users without coding background to promptly access the tools. The tracking tool provided is the most specialised part of the package, as it was developed for particle tracking techniques with a high sampling rate, capable of acquiring long, uninterrupted trajectories, such as Interferometric Scattering (iSCAT).[30] Thereby, the trajectory linking is based on the algorithm that favour strong spatial and temporal connections between consecutive frames. The choice of particle localization algorithm, based on the radial simmetry approach,[31] is linked with the fact that the data which guided the development of the tool features symmetrical, round particles diffusing on a plane. The analysis and simulation tools, on the other hand, have a wider scope of applications. Once the particle trajectories are obtained, either via the provided tracking tool, or imported from other sources, two avenues are available to perform single particle trajectory analysis. The first and more widely adopted, is through the Mean Squared Displacement, which many users will find more familiar and is still conventionally employed. The second avenue, which we named Apparent Diffusion Coefficient analysis, provides more insights into the localisation error and a more intuitive way of monitoring the diffusion behaviour at different time scales. A statistics-driven assists the user in finding the most appropriate diffusion model to describe the particle trajectories. The source code allows more advanced users to customize data analysis, allowing, for example, the introduction of user-defined diffusion models that integrate seamlessly with the proposed analysis pipeline. The track simulator allows the user to generate particle trajectories on a homogeneous plane, or on a surface divided in compartments where the particle can be transiently confined, with a high degree of customisation. We also added the possibility of generating movies from the simulated trajectories with variable levels of contrast, and a user-specified Point Spread Function. This last module can serve as a validation tool of further particle tracking and analysis pipelines.

Methods

Particle tracking

The particle tracking is developed to support a quantitative analysis module which requires extracted trajectories. We introduce a tool which allows users to adjust tracking parameters, preview results and save final trajectories.

The pre-processing of the image sequence is optional and can be applied prior to the tracking algorithm. The tracking is implemented as a two-step process. Firstly, the particles are detected using spot enhancing filter (SEF)[32] and sub-pixel localisation is estimated by the radial symmetry centre approach.[31] Secondly, the detected particles are linked based on their spatial and temporal locations.

The current tool was developed for the iSCAT imaging technique and the pre-processing step is built in accordance to the technique requirement. To distinguish the molecules of interest from the background, the image sequence is divided by a flat field. The flat field represents an undesired static background scattering, and it is calculated by the pixel-wise temporal median filter over a set of frames (1000 frames in the current version). The background is calculated as a temporal average of the entire image sequence and subtracted from each frame. At last, the movie is normalized and brightness adjustment is applied to the image sequence.[33] Although the pre-processing step is developed to subtract a temporal median filter, as it is conventionally done in iSCAT microscopy, it can also be employed for other techniques as a background subtraction tool.

The particle detection exploits a spot enhancing filter (SEF) to enhance the particles and reduce correlated noise in the image. The SEF can be described by the convolution of the original image with a Laplacian-of-Gaussian (LoG) kernel followed by global thresholding to extract the spots.

$$f(x,y,\sigma) = LoG(x,y,\sigma) * g(x,y), \tag{1}$$

where σ is a standard deviation of the LoG. The LoG is a second-order derivative filter, and is commonly used as a base for spot detection.[34] The solution can be adjusted for a different scale by altering σ. The threshold is defined by the average intensity μ_{intens} of the image and its standard deviation σ_{intens}, weighted by the constant c:

$$T = \mu_{\text{intens}} + \sigma_{\text{intens}} \tag{2}$$

The constant c is related to the spot intensity, and together with LoG parameter σ can be adjusted by the user. The local maximum of the image identifies a set of the detected particles in each frame. Further refinement of the particle coordinates (sub-pixel localisation) is implemented with the radial symmetry centre approach.[31] It provides a faster

execution time due to its non-iterative nature, while achieving high accuracy. The size of the region of interest for the sub-pixel localisation and the limitation for the detected peak size can be set manually. The radial symmetry centre approach is limited to a symmetrical round particle, and therefore, for a case of different particle shape, the sub-pixel localisation function has to be replaced with an alternative solution, for example.[35–37] Furthermore, the user should note that this algorithm only provides the centroid positions of the detected particles, and does not feature a precise determination of the standard deviation of the fitted 2D gaussian function, or other parameters such as integrated intensity.

The iSCAT imaging technique provides high temporal resolution. The small displacement of the particle between neighbouring frames allows reducing complexity of the linking algorithm and increase the computational speed. The linking approach focuses on strong spatial connection between frames. Hungarian combinatorial optimisation algorithm[38] is employed for the task of data association. The tolerance of the algorithm towards the spatial and temporal distance can be set by the parameters in the graphical-user interface. The assembled tracks can contain gaps in temporal domain due to the failed detection in one or few frames. These gaps are filled taking into account detections in the neighbouring frames and sub-pixel localisation. At the final stage, all the trajectories are filtered based on their length.

Mean squared displacement analysis

In this and the following section, we will briefly discuss the analysis methods implemented in the TRAIT2D software package. The first method, which we refer to in the code as "MSD analysis", consists of the approach to determine the physical parameters of diffusion dynamics based on the behaviour of the mean squared displacements against the corresponding time interval.

In the case of MSD analysis, the MSD is initially calculated according to the definition:

$$MSD(t_n) = \frac{1}{N-n-1}\sum_{i=1}^{N-n-1} |\underline{r}(t_{i+n}) - \underline{r}(t_n)|^2 \tag{3}$$

whereby $t_n = nt_0$ indicates the n-th time point from the initial time t_0, and $\underline{r}(t_n)$ indicates the two-dimensional position vector at time t_n. Notice that this notation, as well as the entire analysis pipeline hereby implemented, supposes that the time interval between localisation is constant. According to Einstein's theory of Brownian Motion generalized for higher dimensionality,[1,4] this quantity is linked to the diffusion coefficient, D, and time by the formula:

$$MSD(t_n) = 2dDt_n \tag{4}$$

in which n indicates the dimensionality of the system. In the rest of the section, we will assume $d = 2$ to simplify the formalism. In order to account for experimental finite localisation precision, it is necessary to introduce an additive term to 4[39]:

$$MSD(t_n) = 4D(t_n)t_n + 2\delta^2_{x^2+y^2}. \tag{5}$$

The above formulation assumes that the localisation error in the x and y directions is identical. This formula has been generalized to include the possibility of anomalous subdiffusion:

$$MSD(t_n) = 4D(t_n)t_n^{\alpha} + 2\delta^2_{x^2+y^2}. \tag{6}$$

While the user can, of course, fix the localisation error value according to their chosen metric, it is also possible to leave it as a floating parameter. This is theoretically preferable from the point of view that the detection of moving particles is by necessity less precise than that of immobile particles, which are often used as a reference to estimate the localisation precision of a microscopy system. The effect of motion blurring, to which this loss in precision is ascribable, is represented by a negative term by the form[11]:

$$\delta_{Blurring} = -8DRt_{lag} \tag{7}$$

where R is a term which depends on the illumination mode of the chosen microscopy technique, and obeys $0 \leq R \leq 1/4$. It is especially interesting to notice how this term is dependent not on the time variable, but rather to the time resolution of the measurement. As a consequence of this, non-blur-corrected MSD data may suffer from considerable distortion at short time intervals, which coincidentally are the ones that are considered most important in determining the kind of motion the particle undergoes.[12,40] When this term is added to Eq. 5 and Eq. 6, it gives rise to the formulas:

$$MSD(t_n) = 4D(t_n)t_n + 2\delta^2_{x^2+y^2} - 8DRt_0 \tag{8}$$

and

$$MSD(t_n) = 4D(t_n)t_n^{\alpha} + 2\delta^2_{x^2+y^2} - 8DRt_0 \tag{9}$$

which are ultimately used to perform curve fitting and extract the relevant parameters. In the case of free diffusion (5 and 8), we once again remark the necessity to consider an adequate range of data over which to set up the model fitting routine, as highlighted in the ntroduction. The same observation, for the same model, is also valid if the user elects to use the following Apparent Diffusion Coefficient analysis.

Apparent diffusion coefficient analysis

The apparent diffusion coefficient (ADC) analysis differs from the MSD analysis mainly for the central role given to the diffusion coefficient compared to mean squared displacement. We define the ADC as:

$$D_{app}(t_n) = \frac{MSD(t_n)}{4t_n\left(1-\frac{2R}{n}\right)} \tag{10}$$

thereby enabling us to switch to a diffusion coefficient-centric representation. The nomenclature "apparent" is adopted to stress how this quantity is still affected by the localisation error and not the absolute value of the diffusion coefficient. Dividing both terms of Eq. 8 by $4t_n\left(1-\frac{2R}{n}\right)$, and with some elementary algebra, it is possible to derive:

$$D_{app}(t_n) = D(t_n) + \frac{2\delta^2_{x^2+y^2}}{4t_n\left(1-\frac{2R}{n}\right)}. \tag{11}$$

By substituting in place of $D(t_n)$ the appropriate expression for the diffusion coefficient for a specific model, it is therefore possible to obtain a complete model to fit to the apparent diffusion coefficient data, obtained according to Eq. 10. In principle, it is possible to leave the localisation error $\delta^2_{x^2+y^2}$ as a free-floating parameter in the fitting operation, to better estimate the localisation error for moving particles.

The software package hereby presented comes with three built-in diffusion models,[12,41,42] corresponding to free (Brownian) diffusion

$$D(t_n) = D_M, \tag{12}$$

confined diffusion

$$D(t_n) = D_\mu\left(\frac{\tau}{t_n}\right)\left(1-e^{-\frac{\tau}{t_n}}\right), \tag{13}$$

and compartmentalized diffusion, which is the combination of the above two models

$$D(t_n) = D_M + D_\mu\left(\frac{\tau}{t_n}\right)\left(1-e^{-\frac{\tau}{t_n}}\right). \tag{14}$$

The nomenclature for these models follows the convention that D_M is a constant diffusion coefficient, corresponding to theoretical Brownian diffusion, and that τ is the characteristic residence time in the confinement zones described by the confined diffusion model. D_μ is the microscopic diffusion coefficient observed in the aforementioned confinement zones, which can also be expressed as

$$D_\mu = \frac{L^2}{12\tau} \tag{15}$$

where L is the average confinement zone size.[10] It is apparent that while this data analysis pipeline appears more complex, it however readily provides an efficient way to directly extract relevant physical parameters from the particle trajectories. We point out that, although the code we hereby describe comes with the aforementioned built-in models, chosen according to the authors specific research interests, a documented and simple procedure is in place to define a new model, according to the end user-specific wishes. The only restriction is that the custom-defined models can only be used when the analysis module is imported in a Python script. However, the analysis pipeline will integrate this addition seamlessly alongside the other models, or run exclusively on the user-defined models.

The selection of the best model to describe any given track is done by statistical means. Each of the models is fitted to the D_{app} data, and the Bayesian information criterion (BIC) for each model is calculated according to the formula[43]:

$$\mathrm{BIC} = k\ln(n) + n\ln(RSS/n) \tag{16}$$

in which n is the number of data points the model is fitted to, the RSS is the residual sum of squares, and k is the number of degrees of freedom, i.e. the free parameters, of the fit. The most adequate model to describe the diffusion motion is then the one with the lowest value of BIC. Since the BIC is not a goodness-of-fit metric, the analysis function will also perform a one-sided Kolmogorov–Smirnov test, comparing the original ADC data against the results of the fit.

Simulation tools

In order to validate the analysis methods described above, we have developed a simple simulation framework. Our tool can create synthetic particle tracks using various diffusion models. Furthermore, the tracks can be converted into synthetic movies to test the particle detection method.

Track simulation

The free (Brownian) diffusion was simulated by drawing a random walk on a two-dimensional virtual square space of side L, entered by the user. The particle is initialized in the centre of the square. The user also has the ability to select the value for the diffusion coefficient D and the time step dt. The position of the particle is updated at each successive time step in the x and y direction separately:

$$\begin{aligned} x(t+dt) &= x(t) + \sqrt{(2*D*dt)} * (\mathrm{rand}) \\ y(t+dt) &= y(t) + \sqrt{(2*D*dt)} * (\mathrm{rand}) \end{aligned}$$

This set of equations does not strictly respect the condition that $(x(t+dt)-x(t))^2 + (y(t+dt)-y(t))^2 = 4*D*dt$, since doing so would not account for the variability that happens in real simulations. Instead, the theoretical displacement $\sqrt{D*dt}$ is multiplied by a random number ("rand") extracted from a normal distribution of mean $\mu = 0$ and $\sigma = 1$.

The hop diffusion motion was simulated according to the code used for,[44] kindly provided by Dr. J. Keller–Findeisen as a Matlab script, and transcribed into Python by the authors. First of all, a virtual surface of arbitrary dimension is split into an arbitrary number of compartments as a Voronoi diagram with random seed. To each of the compartments, a unique identifier is assigned. A particle is then generated in the centre of the plane, and its motion is simulated in the same way as the Brownian diffusion case. However, an additional condition is in place to simulate the hopping motion. A probability is assigned to the particle of jumping between one compartment and the other, called hopping probability (HP). Every time that the randomly generated displacement would move the particle to a different compartment, a random number between 0 and 1 is extracted. Should this number be less than the value of HP initially fixed, the displacement calculation is repeated in order to keep the particle in the same original compartment.

Confined diffusion can easily be simulated by fixing $HP = 0$.

Movie generation

We have created a simple movie simulator to obtain a synthetic movie from a list of detected or simulated tracks. This simulator is inspired by a similar noisy holographic image simulator.[45] Apart from a mandatory list of tracks to simulate, other, parameters required for the simulation are the spatial (r_{xy}) and temporal (r_t) resolutions, the contrast C = Ip − μbg between the simulated particle intensity Ip and the background signal level (μ_{bg}), and the Gaussian noise level (σ^2_{bg}). Optional inputs are a point-spread function (PSF) stack, which can either be obtained experimentally or estimated with a model of the microscope, such as with DeconvolutionLab2.[46]

The simulator is initialised using the given tracks. The number of time frames, the minimum, and the maximum spot positions in X and Y are extracted from all tracks. These are used to initialise the simulation grid. The (x_{min}, y_{min}) and (x_{max}, y_{max}) positions define the simulation grid width and height. If the grid needs to be square, the min and max between (x_{min}, y_{min}) and (x_{max}, y_{max}) are computed respectively to define the grid shape. The simulation grid is discretized using the spatial resolution r_{xy}. For example, a track whose minimum position is (0,0), and maximum position is (10,10), and for a simulation resolution of $r_{xy} = 1$, will have a shape of 11 × 11, where each grid position represents a pixel of size 1 × 1 μm^2. Furthermore, if the maximum time frame contained in a tract is 10, and if the temporal resolution is set to $r_t = 1$, the simulated movie will be of shape (11 × 11 × 11), where the first two dimensions represent the spatial dimension, and the last is the temporal dimension.

Once the simulation grid is initialised, we first add a background signal $B(x,y,t) \sim \mathcal{N}(\mu_{bg},\sigma_{bg}^2)$ everywhere. Each pixel background intensity follows a normal distribution of mean μbg and of variance σ_{bg}^2, and each simulated background pixels are independent and identically distributed (i.i.d.) random variables. In other words, at this stage there is no spatial or temporal correlation between the pixels intensities. We are using the `util.random_noise` method from the `scikit-image` Python module.[47]

Next, for each spot position in each track, we discretize the position in both the spatial $(x,y) \rightarrow (m_x,m_y)$ and temporal dimensions $(t) \rightarrow (m_t)$ using

$$m_i = \left[\frac{x_i - x_{i,min}}{r_i}\right], \tag{17}$$

where $i \in \{x,y,t\}$ represents the x, y, and t dimension of the spot positions respectively, x_i is its position as recorded in the tract, $x_{i,min}$ is the minimum position within the tract, r_i is the simulation resolution along the i^{th} axis, and $[\cdot]$ represents the rounding, operation to obtain discrete positions m_i within the simulation grid coordinate framework. Each spot position is added iteratively to the simulated movie, and its intensity is given by the desired contrast C for this simulation. At this stage, the simulated movie can be described by the equation

$$M(\mathbf{x}) = (1+SNR)\sum_{i=1}^{N}\delta(\mathbf{x}-\mathbf{m}_i) + B(\mathbf{x}), \tag{18}$$

where $\delta(\mathbf{x} - \mathbf{m}_i)$ is a n-dimensional Dirac delta function, x = (x,y,t) is a (3,1) vector representing the spatiotemporal position within the simulation grid, $\mathbf{m}_i = (m_{x,i},m_{y,i},m_{t,i})$ is the discretized vector position within the simulation grid of the i^{th} spot, and N is the total number of spots among all tracks to simulate. If a PSF is given as input, it will be applied at this stage with a convolution operator. If a 3D PSF stack is given, the central slice of the stack (corresponding to the focus position) will be used instead of the whole 3D stack for the simulation. The PSF can either be an experimentally acquired stack or a simulated, for example, using the external DeconvolutionLab2 plugin for FIJI/ImageJ with the microscope's configuration.[46] To consider the boundary effects, we use zero padding. Thus, the simulated volume is

$$M(\mathbf{x}) = PSF(x,y) \otimes \left[(1+SNR)\sum_{i=1}^{N}\delta(\mathbf{x}-\mathbf{m}_i) + B(\mathbf{x})\right], \tag{19}$$

where $\otimes$ is the convolution product. Finally, i.i.d. Poisson noise is simulated for each pixel separately, using the value of M(x) as the Poisson distribution parameter. The Poisson noise is simulated using the python module `random_noise` from `scikit-image`. Thus we obtain

$$M'(\mathbf{x}) \sim P(k;\lambda(\mathbf{x}) = M(\mathbf{x})), \tag{20}$$

where

$$P(k;\lambda) = \frac{\lambda e^{-\lambda}}{k!} \tag{21}$$

The simulated movie is exported as a tiff stack or an avi file. Figure 3 is a static example of a simulated movie frame using a single simulated track. The image on the left is a movie frame before the PSF convolution and the Poisson noise simulation, and the image on the right is the final simulated movie frame after these operations.

Implementation

TRAIT2D is implemented entirely in Python using available packages.

The software components (simulation, tracking, analysis) have been split into separate modules that have graphical user interfaces (GUIs) available or, in the case of the simulation and analysis libraries, can be imported in Python scripts.

TRAIT2D uses an object-oriented approach for its workflows:

The simulation library uses `Diffusion` objects which hold the parameters and mathematical representation of a diffusion model. Specific models inherit from the base `Diffusion` class. Simulation results are stored in the simulator as a Python dictionary.

In the analysis library, tracks are represented as `Track` objects, which can be created from such a dictionary, a Pandas data frame or a CSV file. Multiple tracks can be pooled in a `ListOfTracks` object. Analysis is executed by calling the corresponding methods on these objects and results are stored in them. Central to the analysis library is, of course, the `.msd_analysis()` and `.adc_analysis()` methods of the `Track` and `ListOfTracks` classes. Through these methods, the user can fit a range of models (included with the library, or custom) to their data. The model fitting itself is ultimately performed using the `.curve_fit()` function of the scipy.optimize module, which uses a nonlinear least squares algorithm to fit the model function to the data. This function returns the optimal value of the parameters, and the relative covariance matrix. The square root of the diagonal elements of said matrix are thus taken as estimator for the fitting error of each parameter.

A unique property of the analysis library is the ability to define and add new diffusion models, through a "singleton" class called `ModelDB`. This functionality is only exposed when importing the analysis module from a Python script but not from the GUI.

Operation

TRAIT2D requires at least a Python 3 (recommended ≥ 3.7) to run. The software also depends on additional packages: Graphical user interfaces are provided through PyQt (analysis) and Tkinter (simulation, tracking). Scipy, matplotlib and pandas are used for model fitting, plotting and loading data, respectively. These and other dependencies are automatically installed when installing from source using the supplied `setup.py` file or from the Python Package Index (PyPI).

Use cases

This section outlines basic use cases for TRAIT2D. The complete documentation as well as a list of tutorials as available at https://eggeling-lab-microscope-software.github.io/TRAIT2D/.

Workflow

The central component of the TRAIT2D software package is the analysis module, through which it is possible to extract physical parameters from the particle trajectories. There are three possible avenues for the user to analyse their data (Figure 1). By using the tracker module, the user can import their timelapses, as TIFF stacks, into the GUI and extract particle trajectories, which can then be exported as a formatted CSV file, ready to be imported in the analysis

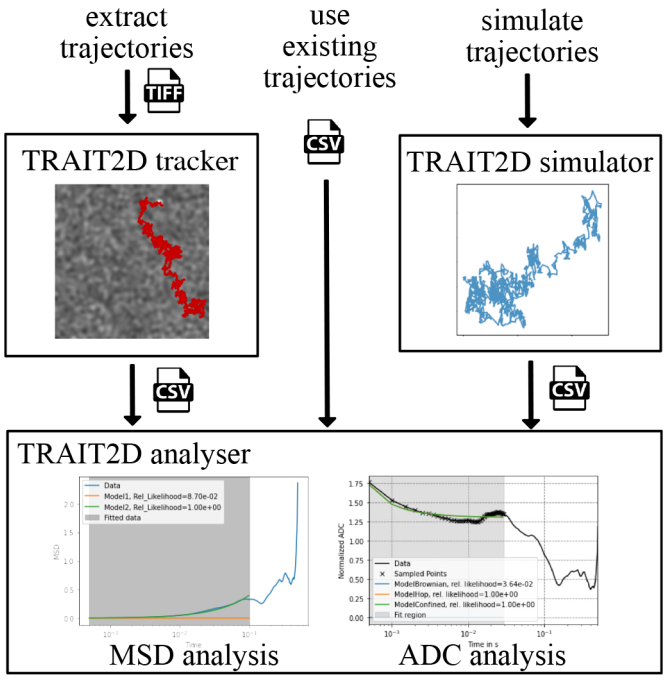

Figure 1. Schematic of the workflow of the TRAIT2D software package, highlighting the three possible paths to feed data to the analysis module: (*left*) through the tracker, starting from a TIFF stack of the diffusing particles, *center* importing trajectories otherwise obtained as a CSV file, (*right*) or by simulating particle trajectories with the simulator module, starting from user-defined parameters.

GUI. Alternatively, if the trajectories are already available to the user through a different particle tracking tool, they can be imported as well, on condition that their formatting is compliant with the format required by the analysis module. Finally, the user could also generate a control dataset through the simulator module, either through the GUI or as a script. The simulated trajectories will then be exported as a properly formatted CSV file, or a compliant data structure in the case of the user script. The Analysis GUI will perform the data analysis following either the MSD or the ADC pipelines (see Methods section), as indicated by the user. We stress that while the GUI exclusively uses the predefined set of models (see Eq. 12, 13 and 14) for the analysis, the module itself allows the addition of more diffusion models as well as batch processing when used in custom scripts, adding an ulterior degree of flexibility.

Compatible file formats and data structures

The analysis library and GUI expects track data to be supplied in a specific format although column names and units are adjustable. At the time of the first release, the trajectories can only be imported from CSV files. A track file must include at least three columns, which contain the localisation time `t` as well as the coordinates `x` and `y`. The order in which these columns appear is not important. A unique numerical identifier `id` must be added as a fourth column in files which contain multiple tracks. This identifier will match each localisation with a specific track.

When working with the analysis library as an imported module inside a Python script, tracks can also be directly loaded from a dictionary or a Pandas data frame as long as they contain the keys described above.

Installation

The package containing the full software can be downloaded directly from PyPI, using the command:

```
pip install trait2d
```

Alternatively, the source code is available at the Github repository:

https://github.com/Eggeling-Lab-Microscope-Software/TRAIT2D.

Graphical user interfaces

Once installed, the different GUIs can be launched by entering `trait2d_analysis_gui`, `trait2d_simulator_gui`, or `trait2d_tracker_gui` in a Python-enabled environment.

Importable modules

The analysis and simulation libraries can be used as modules in Python scripts by importing either `trait2d.analysis` or `trait2d.simulators`. This exposes additional features such as bulk analysis and adding custom diffusion models and allows the user to program pipelines for their specific use case. The tracker software is only available as a GUI. 22

Tracking particles

The particle tracker is available only as a GUI tool at the time of writing. It can be accessed by typing the command `trait2d_tracker_gui` in a terminal with a compatible Python installation. The application accepts 8 bit TIFF stacks as an input. It is possible to run a simple pre-processing step with background subtraction. The tracker GUI provides options to set parameters, and to preview particle detection results frame by frame. Once the parameters for detection and trajectory linking are deemed satisfactory by the user, the application will perform the trajectory linking over the image stack, and export another tiff stack where the detected trajectories are superimposed as a coloured track over the original data (Figure 2). The user can thereafter choose to export the particle tracks as a csv file, with formatting compatible to the analysis tool.

Analysing a track through the MSD pipeline

While the majority of the use cases presented here will be focusing on the more elaborate ADC analysis pipeline, a quick overview of the MSD analysis pipeline will be given in this section as well.

MSD analysis uses only the built-in linear (Eq. 8) and power (Eq. 9) models for fitting, and there is no option to add custom models.

Other features, such as bulk processing, however, are also available for the MSD pipeline and work similar to their ADC counterparts.

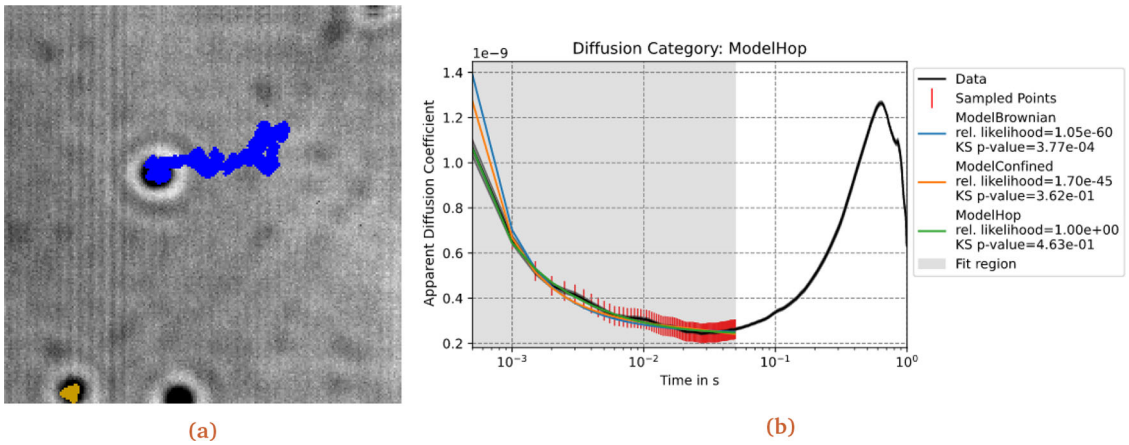

Figure 2. Processing a timelapse image. (a) Example of the output TIFF file, exported by the program after the tracking procedure has been completed. The tracker has found three tracks: one over the full movie length (blue) and a shorter track confined to the sport in the bottom-left corner (yellow). The latter is cut off due to the spot being close to the border. (b) Graphical result of the analysis, through the ADC pipeline, of the track in blue, using the three built-in models, with an opportune analysis interval, initial values and bounds for the parameters.

Below is an example illustrating the process of importing a single track from a file and running MSD analysis on it:

```
from trait2d.analysis import Track
single_track = Track.from_file("track1.csv")
single_track.msd_analysis(fit_max_time=0.5e-1)
```

Once again, we reiterate that the fit_max_time parameter, i.e. the time range over which the MSD analysis is performed, must be chosen with care to avoid misrepresenting the dynamics of the tracked particles, especially in the case of Brownian motion.[16–18]

Analysing a track through the ADC pipeline

In order to be analysed, a track must be contained in a compatible file or data structure (see above). Data analysis can then be performed through either the analysis GUI, accessible by typing `trait2d_analysis_gui` in a terminal with a compatible installation of Python, or through a user script by importing the `trait2d.analysis` module. For the use of the analysis GUI, we would reference the documentation https://eggeling-lab-microscope-software.github.io/TRAIT2D/analysis_gui.html, where it is extensively described. The following code snippets exemplify how to load a single track from one of the example files provided with the code, and analyse it using a Python script.

First of all, the track data needs to be imported as a `trait2d.analysis.Track` object:

```
from trait2d.analysis import Track

single_track = Track.from_file("track1.csv")
```

In order to proceed with the analysis, the models need to be defined. The package comes with three default models (Eq. 12, 13 and 14), which can be handily registered in the `trait2d.analysis.ModelDB` object. All models added to this singleton object will then be used to analyse the track data. Lower and upper bounds, as well as initial values, can also be easily set on a per-model basis. The following code snippet shows, as an example, how to import the built-in brownian diffusion model (Eq. 12), and define the bounds and initial values necessary for the fitting operation:

```
import numpy as np
from trait2d.analysis.models import ModelBrownian
from trait2d.analysis import ModelDB
ModelDB().add_model(ModelBrownian)
```

```
ModelDB().get_model(ModelBrownian).initial = [ 0.0, 0.0]
ModelDB().get_model(ModelBrownian).upper = [ np.inf, np.inf]
ModelDB().get_model(ModelBrownian).lower = [ 0.0, 0.0]
```

For the other models, the procedure is identical, however more or less parameters should be initialized, as required by their expression. Once the `ModelDB` has been populated with the desired diffusion models, the analysis is performed via the command: DM,

```
results = single_track.adc_analysis(fit_max_time=0.5e-1)
```

The analysis module saves the analysis results to the `trait2d.analysis.Track` object, so they can easily be accessed at a later point. More options concerning the whole analysis pipeline are also available in the documentation. Several examples are also provided as downloadable Jupyter Notebooks.

Adding a custom model for ADC analysis

The analysis module allows to define custom diffusion models which can be used in the analysis pipeline, and integrate seamlessly into it. This is done by defining a class which inherits from `trait2d.analysis.models.ModelBase`. This functionality is only available when importing the module in a Python script and not from the GUI.

```
from trait2d.analysis.models import ModelBase
import numpy as np

class MyModelBrownian(ModelBase):
    lower = [ 0.0, 0.0]
    upper = [ np.inf, np.inf]
    initial = [ 0.5e-12, 2.0e-9]

    def __call__(self, t, D, delta):
        return D+delta**2/(2*t*(1-2*self.R*self.dt/t))
```

The newly defined model can be added to `trait2d.analysis.ModelDB`, in addition to or instead of the built-in models and will be used in all subsequent analyses. This functionality is only available when importing the module in a Python script and not from the GUI.

Bulk processing

When using the analysis module inside a Python script, bulk processing of multiple tracks is available. For this, the `trait2d.analysis.ListOfTracks` class is used. In the following example, a CSV file containing multiple tracks is loaded. In addition to MSD and ADC analysis, `ListOfTracks` also exposes useful methods such as `ListOfTracks.plot_trajectories` or `ListOfTracks.adc_summary` to quickly visualise all contents and analysis results of the contained tracks. Bulk processing is also available for MSD analysis *via* `ListOfTracks.msd_analysis`.

```
from trait2d.analysis import ListOfTracks
tracks = ListOfTracks.from_file("three_tracks.csv", unit_length='micrometres')

tracks.plot_trajectories()

from trait2d.analysis.models import ModelBrownian, ModelConfined, ModelHop
from trait2d.analysis import ModelDB

ModelDB().add_model(ModelBrownian)
ModelDB().add_model(ModelConfined)
ModelDB().add_model(ModelHop)

tracks.adc_analysis(fraction_fit_points = 0.15)
tracks.adc_summary(plot_dapp=True, plot_pie_chart=True)
```

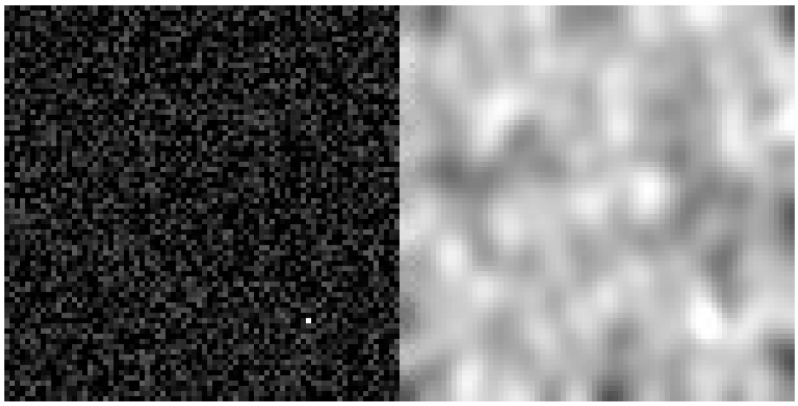

Figure 3. Simulated particle diffusion movie frame. (Left) Frame before and (Right) after the convolution by a PSF and the simulation of Poisson noise. An animated version of this simulation is available in this project Github repository.

Simulating a track

Track simulation can be done either from the GUI, which can be accessed by typing `trait2d_simulator_gui` in a terminal with Python available, or by importing a simulator from `trait2d.simulators`.

The following code example shows the simulation of a track using the imported module and the built-in Brownian diffusion simulator.

```
from trait2d.simulators import BrownianDiffusion

params = dict()
params[ "Tmax"] = 0.5 # Maximum simulation time (s)
params[ "dt"] = 1e-4 # Simulation time resolution (s)
params[ "dL"] = 1e-12 # Simulation spatial resolution (m)
params[ "d"] = 1e-12 # Diffusion coefficient (m^2/s)
params[ "L"] = 1e-5 # Simulation domain size (m)
params[ "seed"] = 42 # Seed to initialize the random generator (for reproducibility)
params[ "quantize"] = False # Do not quantize the position to the spatial grid

simulator_brownian = BrownianDiffusion(**params)

simulator_brownian.run()
```

The GUI also allows for the creation of an image sequence from the simulated track or a loaded trajectory file. Note that this is currently not possible from the imported module inside a Python script. Figure 3 shows such a simulated particle diffusion movie.

Conclusion

TRAIT2D is a Python-based, open source and easily deployable toolkit for the analysis and simulation of two-dimensional SPT data. The combination of particle tracking, simulation, and trajectory analysis in the same package provides a number of benefits for the user. Another beneficial feature of this software is its user-friendliness for inexperienced users, thanks to the graphical user interfaces provided for each module. Nevertheless, the potential for customisation is retained and the scope of the module can be readily expanded by users with more advanced coding background.

Data availability

The tracks used throughout the examples above are available in the source tree at the GitHub repository mentioned above in the `examples` folder: `track1.csv`, `track2.csv`, and `track3.csv` each contain a single track in the format expected by TRAIT2D. `three_tracks.csv` contains the same files which are identified by a separate `id` column. These example files report the trajectories obtained by performing particle tracking on three distinct timelapses of 40 nm diameter, streptavidin-coated gold nanoparticle linked to DSPE-PEG(2000)-biotin lipid analogues diffusing on an artificial membrane (Supported Lipid Bilayer) created on glass support. A `track_example.tif` file, consisting of a movie of a diffusing 40 nm streptavidin-coated gold nanoparticle linked to a biotinylated lipid analogue (DSPE-PEG

(2000)-biotin), is provided as an example for the particle tracker module. The parameters_for_`track_example.csv` provides an optimal set of parameter to track the particle in the aforementioned file, which is directly loadable in the tracker GUI, while the `track_example_output.csv` shows the outcome of the tracking operation.

Software availability

- Software available from https://github.com/Eggeling-Lab-Microscope-Software/TRAIT2D/releases/ and https://pypi.org/project/trait2d.
- Source code available from https://github.com/Eggeling-Lab-Microscope-Software/TRAIT2D.
- Archived source code at time of publication: https://doi.org/10.5281/zenodo.4725268.
- License: GPL-3.0 License.

Acknowledgements

FR and CE would like to acknowledge Dr. B. Christoffer Lagerholm for his invaluable scientific advice. The authors would like to thank Dr. J. Keller-Findeisen for allowing the translation of the original algorithm for compartmentalised diffusion simulation.

References

1. Einstein A: **On the Motion of Small Particles Suspended in a Stationary Liquid, as Required by the Molecular Kinetic Theory of Heat.** *Annalen der Physik.* 1905; **322**: 549–560. 1521-3889.
Publisher Full Text | Reference Source

2. von Smoluchowski M: **Zur kinetischen Theorie der Brownschen Molekularbewegung und der Suspensionen.** *Annalen der Physik.* 1906; **3260**(14): 756–780. 00033804.
Publisher Full Text | Reference Source

3. Clausen MP, Lagerholm BC: **The probe rules in single particle tracking.** *Curr Protein Pept Sci.* dec 2011; **120**(8): 699–713. 1875-5550.
Publisher Full Text | Reference Source

4. Manzo C, Garcia-Parajo MF: **A review of progress in single particle tracking: from methods to biophysical insights.** Dec 2015.
Reference Source

5. Perrin J: *Brownian Movement and Molecular Reality.* London: Taylor & Francis; 1910.

6. Umemura YM, Vrljic M, Nishimura SY, *et al.*: **Both MHC class II and its GPI-anchored form undergo hop diffusion as observed by single-molecule tracking.** *Biophys J.* 2008; **950**(1): 435–450. 15420086.
PubMed Abstract | Publisher Full Text | Free Full Text

7. Balzarotti F, Eilers Y, Gwosch KC, *et al.*: **Nanometer resolution imaging and tracking of fluorescent molecules with minimal photon fluxes.** *Science.* feb 2017; **3550**(6325): 606–612. 0036-8075.
PubMed Abstract | Publisher Full Text | Reference Source

8. Taylor RW, Sandoghdar V: *Interferometric Scattering (iSCAT) Microscopy and Related Techniques.* Cham: Springer International Publishing; 2019; pages 25–65. 978-3-030-21722-8.
Publisher Full Text

9. Saxton MJ: **Single-particle tracking: models of directed transport.** *Biophy J.* 1994; **670**(5): 2110–2119. 00063495.
PubMed Abstract | Publisher Full Text | Free Full Text

10. Wieser S, Moertelmaier M, Fuertbauer E, *et al.*: **(Un)confined diffusion of CD59 in the plasma membrane determined by high-resolution single molecule microscopy.** *Biophy J.* 2007; **920**(10): 3719–3728. 00063495.
PubMed Abstract | Publisher Full Text | Free Full Text

11. Michalet X: **Mean square displacement analysis of single-particle trajectories with localization error: Brownian motion in an isotropic medium.** *Physical Review E.* oct 2010; **820**(4): 041914. 1539-3755.
PubMed Abstract | Publisher Full Text | Free Full Text | Reference Source

12. Goulian M, Simon SM: **Tracking Single Proteins within Cells.** *Biophys J.* oct 2000; **790**(4): 2188–2198. 0006-3495.
PubMed Abstract | Publisher Full Text | Free Full Text | Reference Source

13. Christoffer Lagerholm B, Andrade DM, Clausen MP, *et al.*: **Convergence of lateral dynamic measurements in the plasma membrane of live cells from single particle tracking and STED-FCS.** *J Phys D Appl Phys.* 2017; **500**(6): 063001. 0022-3727.
PubMed Abstract | Publisher Full Text | Free Full Text | Reference Source

14. Martin DS, Forstner MB, Käs JA: **Apparent subdiffusion inherent to single particle tracking.** *Biophy J.* 2002; **830**(4): 2109–2117. 00063495.
PubMed Abstract | Publisher Full Text | Free Full Text

15. Sako Y, Kusumi A: **Barriers for Lateral Diffusion of Transferrin Receptor in the Plasma Membrane as Characterized by Receptor Dragging by Laser Tweezers: Fence versus Tether.** *J Cell Biol.* 1995.
PubMed Abstract | Publisher Full Text | Free Full Text | Reference Source

16. Berglund AJ: **Statistics of camera-based single-particle tracking.** *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* Jul 2010; **82**(1): 1–8.
PubMed Abstract | Publisher Full Text

17. Michalet X, Berglund AJ: **Optimal diffusion coefficient estimation in single-particle tracking.** *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* jun 2012; **85**(6): 1–14.
PubMed Abstract | Publisher Full Text

18. Vestergaard CL, Blainey PC, Flyvbjerg H: **Optimal estimation of diffusion coefficients from single-particle trajectories.** *Phys. Rev. E Stat. Nonlinear Soft Matter Phys.* feb 2014; **89**(2): 022726.
PubMed Abstract | Publisher Full Text

19. Chenouard N, Smal I, de Chaumont F, *et al.*: **Objective comparison of particle tracking methods.** *Nat. Methods.* 2014; **11**(3): 281–9.
PubMed Abstract | Publisher Full Text | Reference Source

20. Tinevez JY, Perry N, Schindelin J, *et al.*: **TrackMate: An open and extensible platform for single-particle tracking.** *Methods.* 2017; **115**: 80–90.
PubMed Abstract | Publisher Full Text

21. Stein SC, Thiart J: **TrackNTrace: A simple and extendable open-source framework for developing singlemolecule localization and tracking algorithms.** *Sci. Rep.* Dec 2016; **6**(November): 37947.
PubMed Abstract | Publisher Full Text

22. Lund FW, Jensen MLV, Christensen T, *et al.*: **SpatTrack: An imaging toolbox for analysis of vesicle motility and distribution in living cells.** *Traffic.* 2014; **15**(12): 1406–1429.
PubMed Abstract | Publisher Full Text

23. Granik N, Weiss LE, Nehme E, *et al.*: **Single-particle diffusion characterization by deep learning.** *Biophys. J.* 2019; **117**(2): 185–192.
PubMed Abstract | Publisher Full Text | Reference Source

24. Arts M, Smal I, Paul MW, *et al.*: **Particle mobility analysis using deep learning and the moment scaling spectrum.** *Sci. Rep.* 2019; **9**: 17160.
PubMed Abstract | Publisher Full Text

25. Muñoz-Gil G, Volpe G, Garcia-March MA, *et al.*: **Objective comparison of methods to decode anomalous diffusion.** *Nat. Commun.* 2021; **12**(1): 6253.
PubMed Abstract | Publisher Full Text

26. Thapa S, Lomholt MA, Krog J, *et al.*: **Bayesian analysis of single-particle tracking data using the nested-sampling algorithm: maximum-likelihood model selection applied to stochastic-diffusivity data.** *Phys. Chem. Chem. Phys.* 2018; **20**(46): 29018–29037.
Publisher Full Text

27. Hansen AS, Woringer M, Grimm JB, *et al.*: **Robust model-based analysis of single-particle tracking experiments with spot-on.** *elife.* 2018; **7**: 1–33.
PubMed Abstract | Publisher Full Text

28. Schindelin J, Arganda-Carreras I, Frise E, *et al.*: **Fiji: An open-source platform for biological-image analysis.** *Nat Methods.* 2012; **90**(7): 676–682. 15487091.
Publisher Full Text

29. De Chaumont F, Dallongeville S, Chenouard N, *et al.*: **Icy: An open bioimage informatics platform for extended reproducible research.** *Nat Methods.* 2012; **90**(7): 690–696. 15487091.
PubMed Abstract | Publisher Full Text

30. Taylor RW, Sandoghdar V: **Interferometric Scattering (iSCAT) Microscopy & Related Techniques.** *ArXiv.* dec 2018; pages 1–42.
Reference Source

31. Parthasarathy R: **Rapid, accurate particle tracking by calculation of radial symmetry centers.** *Nat. Methods.* 2012; **9**(7): 724–726.
PubMed Abstract | Publisher Full Text

32. Smal I, Loog M, Niessen W, *et al.*: **Quantitative comparison of spot detection methods in fluorescence microscopy.** *IEEE Trans. on Medical Imaging.* 2010; **290**(2): 282–301.
PubMed Abstract | Publisher Full Text

33. Ortega-Arroyo J, Cole D, Kukura P: **Interferometric scattering microscopy and its combination with single-molecule fluorescence imaging.** *Nat Protocols.* 2016; **110**(4): 617–633. 1754-2189.
PubMed Abstract | Publisher Full Text | Reference Source

34. Han KTM, Uyyanonvara B: **A survey of blob detection algorithms for biomedical images.** *7th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES).* 2016; pages 57–60.

35. Mortensen KI, Churchman LS, Spudich JA, *et al.*: **Optimized localization analysis for single-molecule tracking and super-resolution microscopy.** *Nat. Methods.* 2010; **7**(5): 377–381.
PubMed Abstract | Publisher Full Text

36. Andersson SB: **Localization of a fluorescent source without numerical fitting.** *Opt. Express.* 2008; **16**(23): 18714–18724.
PubMed Abstract | Publisher Full Text

37. Cheezum MK, Walker WF, Guilford WH: **Quantitative comparison of algorithms for tracking single fluorescent particles.** *Biophys. J.* 2001; **81**(4): 2378–2388.
Publisher Full Text0006-3495

38. Kuhn HW: **The hungarian method for the assignment problem.** *Naval Res Logistics Quarterly.* 1955; **20**(1-2): 83–97.
Publisher Full Text

39. Dietrich C, Yang B, Fujiwara T, *et al.*: **Relationship of lipid rafts to transient confinement zones detected by single particle tracking.** *Biophy J.* 2002; **820**(1): 274–284.
PubMed Abstract | Publisher Full Text | Free Full Text

40. Qian H, Sheetz MP, Elson EL: **Single particle tracking Analysis of diffusion and flow in two-dimensional systems.** *Biophy J.* 1991; **600**(4): 910–921.
PubMed Abstract | Publisher Full Text | Free Full Text

41. Jacobson K, Liu P, Lagerholm BC: **The Lateral Organization and Mobility of Plasma Membrane Components.** *Cell.* 2019; **1770**(4): 806–819.
Publisher Full Text

42. Destainville N, Saulière A, Salomé L: **Comment to the article by Michael J. Saxton: A biological interpre-tation of transient anomalous subdiffusion. I. Qualitative model.** *Biophys. J.* 2008; **95**(7): 3117–3119.
PubMed Abstract | Publisher Full Text

43. Burnham KP, Anderson DR: *Model Selection and Multimodel Inference: A practical Information-Theoretic-Approach.* Springer; 2002. 0387953647.
Reference Source

44. Andrade DM, Clausen MP, Keller J, *et al.*: **Cortical actin networks induce spatio-temporal confinement of phospholipids in the plasma membrane – a minimally invasive investigation by STED-FCS.** *Sci Rep.* 2015; **50**(May): 11454. 2045-2322.
PubMed Abstract | Publisher Full Text | Free Full Text | Reference Source

45. Douglass KM: **Modeling noise for image simulations.** 2017.
Reference Source

46. Sage D, Donati L, Soulez F, *et al.*: **DeconvolutionLab2: An open-source software for deconvolution microscopy.** *Methods.* feb 2017; **115**: 28–41. 10462023.
PubMed Abstract | Publisher Full Text | Reference Source

47. Van Der Walt S, Schönberger JL, Nunez-Iglesias J, *et al.*: **Scikit-image: Image processing in python.** *PeerJ.* 2014; **20140**(1): 1–18. 21678359.
Publisher Full Text

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 2

Reviewer Report 09 February 2022

https://doi.org/10.5256/f1000research.119173.r121719

✔ **Xavier Michalet**

[1] Department of Chemistry and Biochemistry, University of California, Los Angeles (UCLA), Los Angeles, CA, USA

[2] California NanoSystems Institute, University of California at Los Angeles, Los Angeles, CA, USA

The authors have addressed the comments of my initial review, further increasing the value of their work. I have only a few additional comments, one about the science itself, the others about the new software version.

1. Optimal number of points for MSD fits:

In their response, the authors write: "We acknowledge the fact that in the paper pointed out by the reviewer a rough way to estimate the optimal number of points is provided, which would prove useful in the analysis of Brownian diffusion". That work does indeed provide a heuristic formula for the optimal number of points as a convenience, but the exact values are in the paper for everyone to peruse (they could in fact be stored as a precomputed array of values and change points), or they can be recalculated as described. The most important point is that prior knowledge of the relevant parameters (D and the localization uncertainty, sigma) is not needed to use it if the associated iterative numerical algorithm is used. A MATLAB script implementing this algorithm (as well as the unrelated but just as useful MLE approach introduced by Berglund) is provided as supporting information to this paper. There shouldn't be any difficulty translating it in Python.

2. Software tests:

- 1 - trait2d_simulator_gui

After fixing a number of Python modules dependency errors (a step that requires minimal Python proficiency to overcome, but would surely discourage an unprepared user), I ran trait2d_simulator_gui and tried generating a trajectory using the default parameters (hopping diffusion). The file trait2d_simulator_gui error 1.txt reproduces the error message sent to the terminal. Similar errors were generated trying to simulate a "free diffusion" trajectory or a full image sequence.

The error seems to reside in the self.seed=int(self.param_seed.get()) call and is apparently due to the default seed value, 'None'. Using any integer value fixes the problem.
As a note, generating an image sequence using the default parameters appears to require quite a bit of RAM and time on the fairly recent and powerful laptop I was using.

- 2 - trait2d_analysis_gui

After fixing a bunch of additional dependency errors, I was able to test the GUI using one of the trajectory files (.csv) in the examples folder. While the GUI seemed to be working fine, a bunch of additional errors were outputted in the terminal window (see trait2d_analysis_gui error 2.txt file).

- 3 - trait2d_tracker_gui

Once again, I had to fix some dependency issues before successfully launching the app. The example file was processed successfully using the default parameters.
Summary:

- The installation problems I ran into are not insurmountable assuming some basic familiarity with the pip command but could result in a lot of overhead for the authors if they target non-Python savvy users. It might be useful to release some "standalone" versions (maybe PC and Mac only?) of the runtime to avoid these problems.

- I found the color contrast rather poor for the simulator and tracker GUIs. Consider using the same color scheme as for the analysis GUI for instance, although the graph part of it is also lacking contrast.

3. As a last point, I would suggest setting the correct option for ReadTheDocs such that a PDF version of the manual can be downloaded (web navigation is not always the best approach to discover a specific feature of a new software).

Content of trait2d_simulator_gui error 1.txt:

```
c:\xxxxx\trait2d test\lib\site-packages\trait2d\simulators.py:279: UserWarning: mean displacement (2D) is more than half of average compartment length, results might be compromised
  warnings.warn("mean displacement (2D) is more than half of average compartment length, results might be compromised")
Exception in Tkinter callback
Traceback (most recent call last):
  File "c:\xxxxx\trait2d test\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "c:\xxxxx\trait2d test\lib\site-packages\trait2d_simulator_gui\__init__.py", line 304, in generate_trajectory
    self.read_parameters()
  File "c:\xxxxx\trait2d test\lib\site-packages\trait2d_simulator_gui\__init__.py", line 391, in read_parameters
    self.seed=int(self.param_seed.get())
ValueError: invalid literal for int() with base 10: 'None'
```

Content of trait2d_analysis_gui error 2.txt:

```
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:22: RuntimeWarning: divide by
zero encountered in true_divide
  return D + delta**2 / (2*t*(1-2*self.R*self.dt/t))
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:22: RuntimeWarning: invalid
value encountered in multiply
  return D + delta**2 / (2*t*(1-2*self.R*self.dt/t))
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:34: RuntimeWarning: divide by
zero encountered in true_divide
  return D_micro * (tau/t) * (1 - np.exp(-t/tau)) + \
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:34: RuntimeWarning: invalid
value encountered in multiply
  return D_micro * (tau/t) * (1 - np.exp(-t/tau)) + \
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:35: RuntimeWarning: divide by
zero encountered in true_divide
  delta ** 2 / (2 * t * (1 - 2 * self.R * self.dt / t))
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:35: RuntimeWarning: invalid
value encountered in multiply
  delta ** 2 / (2 * t * (1 - 2 * self.R * self.dt / t))
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:48: RuntimeWarning: divide by
zero encountered in true_divide
  D_micro * (tau/t) * (1 - np.exp(-t/tau)) + \
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:48: RuntimeWarning: invalid
value encountered in multiply
  D_micro * (tau/t) * (1 - np.exp(-t/tau)) + \
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:49: RuntimeWarning: divide by
zero encountered in true_divide
  delta ** 2 / (2 * t * (1 - 2 * self.R * self.dt / t))
c:\xxxxx\trait2d test\lib\site-packages\trait2d\analysis\models.py:49: RuntimeWarning: invalid
value encountered in multiply
  delta ** 2 / (2 * t * (1 - 2 * self.R * self.dt / t))
```

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Single-molecule biophysics, single-photon detectors, fluorescence spectroscopy and imaging

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 05 October 2021

https://doi.org/10.5256/f1000research.58301.r95432

Eva Christensen Arnspang
Department of Chemical Engineering, Biotechnology and Environmental Technology, University of Southern Denmark, Odense, Denmark

In this paper, the authors describe a novel analysis code for analyzing SPT data allowing for short-term observations. The code allows for tracking even at fast acquisition rates, yields information about different diffusion modes, and includes a GUI. A very well-written paper on novel SPT analysis method. I should be very interested in running the code in my own lab.

The paper in its current form is acceptable for indexing, but it would enhance the reading experience for the reader with few well-chosen image examples of how the different components in the code improve the image analysis added, e.g. blurring, in addition to Figures 1 and 2. I would also recommend citing Mortensen *et al*., 2010[1] - in this paper, a similar analysis technique is described.

References

1. Mortensen KI, Churchman LS, Spudich JA, Flyvbjerg H: Optimized localization analysis for single-molecule tracking and super-resolution microscopy.*Nat Methods*. 2010; **7** (5): 377-81 PubMed Abstract | Publisher Full Text

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioimaging, membrane proteins, nano domains, cell organelles, super-

resolution microscopy

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 21 Dec 2021

Francesco Reina, Leibniz-Institut für Photonische Technologien e.V, Jena, Germany

We thank the reviewer for this kind report. We look forward to the applications of our code in her research group and remain open to suggestions for further changes, optimizations, and support requests.

We are not sure what the reviewer intends when suggesting how the code would improve image analysis. In fact, the code provided does not act on the time-lapse image stacks themselves in any part, save from a simple background subtraction pre-processing step in the tracker GUI, and does not add any blurring on top. The blurring correction added refers to the fact that the localization of a diffusing particle, when imaged in its motion, will be subject to additional errors, which can be corrected with a term dependent on the sampling rate of the measurement, the diffusion coefficient, and a coefficient R dependent on the illumination mode used by the optical setup [1]. In case the reviewer intended something else, we could provide some examples upon suggestion.

Concerning the additional reference, we appreciate the suggested work, which is definitely a worthwhile read for every scientist seeking a comparison of methods for accurate single-particle localization. Therefore, we have added the suggested reference with appropriate text modifications.

Competing Interests: No competing interests were disclosed.

Reviewer Report 22 September 2021

https://doi.org/10.5256/f1000research.58301.r93050

?

Xavier Michalet
[1] Department of Chemistry and Biochemistry, University of California, Los Angeles (UCLA), Los Angeles, CA, USA
[2] California NanoSystems Institute, University of California at Los Angeles, Los Angeles, CA, USA

Summary:

This manuscript by Reina *et al*. describes a Python package implementing 1) single-particle

tracking (SPT); 2) single-trajectory analysis (STA) by MSD (isotropic Brownian diffusion and isotropic anomalous diffusion) or ADC (Apparent Diffusion Coefficient, an acronym coined by the authors to designate a single-MSD point estimate of the diffusion coefficient); and 3) SPT movie simulation. The package is provided as a series of Python modules installed using the pip install command and can be run as 3 separate GUIs, as Python command lines, or within Jupyter Notebooks. The package comes with fairly detailed documentation (ReadTheDocs format, with code comments used to describe the different functions and classes in the module, and additional illustrated text description of the installation procedure and basic tutorials).

The manuscript consists of a very basic introduction on the basics of SPT and single-trajectory analysis and a brief overview of a few examples of how to use the 3 different modules (SPT, Analysis and Simulation), and provides the necessary links to the Github repository and documentation. It is fairly well written and is sufficient to assess what the package will allow an inexperienced user to do. Basic attempts by this reviewer to test the different GUIs, followed by some failures or lengthy lists of warnings, seem to indicate that minor additional details and guidance could be useful for potential inexperienced users. The samples of code checked by the reviewer appear well-documented, but an overview of the relationship between different functions and classes might be useful for developers interested in using this project as a basis for their own tools (contributions do not appear to be discussed in the Github repository).

The following are a few remarks which the authors could potentially use to remedy these deficiencies.

Introduction

While this is not a review article, it seems that the introduction ignores a large body of literature and code on SPT and STA. For instance, the Particle Tracking Challenge paper (2014)[1] could be cited as it was an interesting effort to compare the multitude of isolated tools existing at the time, and would have provided a convenient set of test files to run the present software package on. There are many (literally dozens of) freely available packages to track and analyze single-particle data (easily found by an internet search with single-particle tracking as the keyword) and not all are using Matlab or “require licensed software to be executed” (Matlab can usually be replaced by the open-source Octave package), some are web-based (Spot-On) and many come with GUIs; therefore, some of the arguments put forward by the authors to justify their effort are somewhat artificial. This does not minimize the appreciable effort to document and simplify access to this type of tool for the novice user, although some additions might be needed (see later, GUI tests).

Technically, the paper only scratches the surface of SPT and STA, but for obvious reasons since it is primarily a software presentation manuscript. However, since it appears, at least in part, targeted to novice SPT users, it would seem important to emphasize some important simplifications used in its algorithm implementations. For instance, the authors use the “Radial symmetry center approach” (which I was not aware of) because of its speed, but it is assuming radial symmetry, which might not always be appropriate (e.g. in the case of double-helix or astigmatic PSFs or simply long integration combined with fast diffusing particles). From a coding perspective, it could be useful to document how to replace this part of the code with another (or better yet, provide an example of such an optional alternative fitting algorithm – FluoBancroft comes to mind).

When mentioning the blurring effect on diffusion analysis, the paper by Goulian & Simon is

seminal, but in addition to Michalet 2010 already cited, Berglund 2010[2], Michalet & Berglund 2012 [3] and Vestergaard 2012[4] went some extra length to show how this affected the uncertainty on estimated parameters when performing optimal analysis (a question addressed further below). The authors might find it useful to point their readers to these papers (or not: it is not my intention to be self-serving, only to point out the often forgotten and rediscovered fact that this is not a completely trivial matter), which are a good starting point to follow the more recent literature on parameter estimation in more complex models.

Trajectory analysis

MSD analysis, even with a large number of locations, is subtle (as just mentioned, see e.g. Fig. 7 of Michalet & Berglund 2012[3]), in the sense that the number of MSD points used has an effect on the precision of the extracted parameters. In fact, there is an "optimal" number (or range of numbers) to be used if one wants to minimize the error on the estimated parameters, and to some extent, this number can be automatically determined. Is that number the "Range" parameter found at the bottom of the Single-Track Analysis GUI window, and in that case, could an optional determination be offered? How is the user supposed otherwise to guess the best value without any theoretical insights?

Anomalous diffusion analysis in the presence of an error is treated only in an ad-hoc manner in Eq. (8) (I could not find mention of it in ref. [30]) and it is therefore not quite clear how accurate the determination of either error, D_alpha or exponent alpha, is (not mentioning the eluded optimal number of points to use in the analysis). The point is, as discussed extensively in the recent literature, that MSD is almost certainly not an optimal method to analyze anomalous diffusion trajectories. I will not comment on the ADC approach, but clearly, the same comments would apply (optimal number of points and uncertainty on extracted parameters). In this respect, the "error" outputs on fitted parameters provided by the STA module are a bit mysterious (it would be a simple addition to write down their definitions in an appendix).

Tracking

I would likewise recommend discussing (or cite a reference for) the LoG kernel and what its role is in image analysis. There are two user parameters to specify when implementing this kernel and it is unclear how to best choose them. In fact, the same question arises for the radial symmetry center algorithm (2 parameters) and the connection algorithm (2 parameters).

All the previous discussions about tracking uncertainties, SNR, etc., seem for naught when contemplating the CSV file format for SPT trajectories, as it doesn't include fields such as integrated intensity or localization uncertainty. Clearly, these parameters should be outputs of the SPT part and could/should be used to take into account as much possible information on the data as possible (for instance, to set the localization uncertainty to a fixed value?).

Simulations

A priori, the correct simulation of diffusion with coefficient D is to use displacements distributed as N(0,sqrt(2D dt)) (see e.g. SI of ref. 10). The authors use instead sqrt(D dt) N(0, 1), which appears different based on the linearity of normal distributions (aN(0,s) + b = N(b,|a|s)). Please clarify.

On P7, provide an explicit definition of the SNR, as there are several ways to define something that can play the role of SNR.

Test of the provided code

I only tested the GUIs after installation in a dedicated Anaconda environment. Unfortunately, I encountered a few issues, which the authors might be able to address in the documentation.

trait2d_analysis_gui: Tested on the provided trackn.csv files, the terminal window spits out a large number of warnings.

- How is the user supposed to deal with those (and should they care)?
- What is the "Max. fit iterations" parameter for?
- What is the "Use parameters as initial guesses" checkbox for?
- What is the "Range" parameter for (it seems to be used to determine the number of data points (MSD or ADC) used for analysis)? I am assuming this is equivalent to the "fit_max_time" parameter discussed in the command line version (P 10) but again, this could be leaving a lot of freedom to a user unaware of the literature.

Trait2d_simulator_gui: Tested with default parameters and "Free Diffusion" resulted in errors in the Terminal window (for both Generate and Show Track). Same with Hopping Diffusion. It would be useful if the default parameters actually resulted in usable data. Minor tinkering resulted in viable results, but the generation of a simulated movie can take a considerable amount of time (and doesn't seem to be parallelized); therefore a progress bar or some indication of the current frame number could be helpful. An indication of the resulting movie size could also be useful beforehand (those files can grow rapidly large when using the wrong parameters).

Trait2d_tracker_gui: It would be nice to provide an image stack example (and 8-bit TIFF stacks might be limiting in a world where most cameras output 16-bit images). I used the test image stack provided with TrackMate (a FIJI plugin) and the default parameters (with Light Spot option) but that failed (empty track). The file I generated by simulation using the previous module worked somewhat but it was difficult to figure out optimal parameters, and once tracking is started, it appears impossible to abort it (the progress was clearly indicated by the terminal output). Enabling partial initial processing of large stacks could prevent this kind of problem. Likewise, providing a short example stack – for instance, that discussed in the text – and initializing the GUI with appropriate parameters could help a user play around with parameters and understand the influence of each of the 7 important ones. Saving/loading analysis parameters for future reuse with similar datasets could also be useful.

Typos:

- "otherDM", "roundingDM", "diffusion models DM", ""command: DM" throughout the text.
- The sentence "allows the user to rely on objective parameters which diffusion model is more appropriate" does not make sense.
- P4: elucidate >> discuss.

- P6: The sentence “the condition that (x(t + dt)-x(t))^2 + (y(t + dt)-y(t))^2,..." does not make sense.
- P8: stored to the simulator >> stored in the simulator
- can be batched >> can be pooled
- stored to them >> stored in them
- at least a Python 3 >> at least Python 3
- P10: compatible to the analysis tool >> compatible with the analysis tool
- References: some of the journal volume numbers have 3 or 4 digits, which appears to be a mistake.

References

1. Chenouard N, Smal I, de Chaumont F, Maška M, et al.: Objective comparison of particle tracking methods.*Nat Methods*. 2014; **11** (3): 281-9 PubMed Abstract | Publisher Full Text
2. Berglund AJ: Statistics of camera-based single-particle tracking.*Phys Rev E Stat Nonlin Soft Matter Phys*. 2010; **82** (1 Pt 1): 011917 PubMed Abstract | Publisher Full Text
3. Michalet X, Berglund AJ: Optimal diffusion coefficient estimation in single-particle tracking.*Phys Rev E Stat Nonlin Soft Matter Phys*. 2012; **85** (6 Pt 1): 061916 PubMed Abstract | Publisher Full Text
4. Vestergaard CL, Blainey PC, Flyvbjerg H: Optimal estimation of diffusion coefficients from single-particle trajectories.*Phys Rev E Stat Nonlin Soft Matter Phys*. 2014; **89** (2): 022726 PubMed Abstract | Publisher Full Text

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Single-molecule biophysics, single-photon detectors, fluorescence spectroscopy and imaging

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 21 Dec 2021

Francesco Reina, Leibniz-Institut für Photonische Technologien e.V, Jena, Germany

We thank the reviewer for this extensive and constructive report. We acknowledged his comments, and have accordingly improved the manuscript, code, and documentation. Herein, we address all the reviewer's concerns.

Introduction

We appreciate the time and effort spent by the reviewer to analyse the introductory paragraph. Given the extensive body of literature on SPT data analysis, and with the good intention of not expanding the introductory paragraph too much, indeed some literature has been glossed over. We have included the Particle Tracking Challenge paper among the references, for a more comprehensive overview of work already carried out on SPT and STA data analysis.

On the topic of why the tool has been developed, the reviewer's comment has sparked a constructive discussion among the authors, which led to a reframing of the tool in the context of Single Particle Tracking accessibility and the tools already available. This now reflects more accurately the intentions of the authors, and we thank the reviewer for their honesty that allowed us to further consider this point.

We have highlighted limitations of the radial symmetry centre method and the reason it was selected among other existing methods. We also have proposed alternative solutions, providing general direction to replace the current sub-pixel localisation function in the code.

We have chosen to address the choices of algorithm implementations as pointed out by this reviewer. In particular, as mentioned by the reviewer, we specifically addressed the choice to adopt the Radial Symmetry Center approach in order to fit the PSF of the diffusing particle in the movie as a deliberate decision originated by the research environment and the kind of data that favoured the inception of the present tool. Nevertheless, together with a general invitation to contribute to the code, a roadmap on how to implement a different tracking algorithm is provided.

The short but comprehensive list of works suggested by the reviewer has been taken into account, and an invitation to further reading on the appropriate amount of points to take into consideration for the analysis has been added as a precaution to the correct use of the tool.

Trajectory Analysis

Naturally, we concur with the reviewer that the number of MSD points, and indeed the model fitting methodology, have a significant effect on the precision of the parameters extracted, as this has been derived in several papers mentioned thus far in the reviewer report. The “Range” parameter, in the present tool, is a visual aid for the user to keep track of the time range considered in the model fitting. We acknowledge the fact that in the paper pointed out by the reviewer a rough way to estimate the optimal number of points is provided, which would prove useful in the analysis of Brownian diffusion especially, however, we deem this to be beyond the scope of this initial release. Nevertheless, given the open-source nature of the project, a contribution in this direction could also be implemented in the main code, and we rendered this process easier by adding specific guidelines for contribution (whose previous lack we thank the reviewer for pointing out). We, therefore, invite the reviewer to contribute an issue to our GitHub once the review process is complete so that the topic can be further debated and the tool developed in this direction.

The inclusion of Ref [30] before equation (5) is an oversight by the authors and it has been amended, we apologize for the confusion. The inclusions of Eq. (5) and (8) are included as the MSD is still widely used as a measure to at least qualitatively evaluate anomalous diffusion [2], and thus its inclusion was deemed worthwhile. It should be noted, however, that while in the GUI, in the present release, both Eq. (7) and (8) are always fitted to the experimental data, this is not the case for a customized script including our library, which allows the user to freely exclude either, or define their own models altogether.

Insofar as the accuracy of the determination of the parameters is concerned, we have added statements in the “Implementation” section detailing which algorithms in the SciPy library are employed and how the errors are thus calculated. Regarding the absolute accuracy of the method, we have emphasized the need for the user to further explore the topic, and we are sure that the present report, being publicly available, will also provide further guidance to novice SPT adopters.

The comment on the ADC analysis regarding the errors of the estimated parameters was resolved by commenting on the model fitting procedure as for the MSD analysis. A reference to the range as intended by the reviewer is further added for the case of the free diffusion, in Eq. (11), since in this case it clearly would apply.

Tracking

The reference to the Laplacian-of-Gaussian and a short discussion of its role is included in the manuscript, as well as the influence of the LoG kernel and threshold parameters into the spot detection. Suggestions on the parameter settings are also provided in the software manual. We have added a link to the manual in the software interface to make it convenient for the user to find the information.

We agree with the reviewer that localization uncertainty and integrated intensity can be a useful addition to the tracker output, and as a GitHub-based open-source project, we intend to extend the current version with a contribution from the community and our team. Regarding the current tool, the Radial Symmetry center approach in the implementation made available as a reference in the text only provides the standard deviation of the fitted

2D gaussian as an estimation, and not a calculated parameter, and does not include an integrated intensity. However, for the purposes of simple centroid determination, it provides an accurate and fast avenue. A cautionary statement regarding the use of the tool has been added to make the users aware of these characteristics of the tool.

Simulations
The displacement distribution was modified to use N(0,sqrt(2D dt)). numpy.random.randn returns a sample from the standard normal distribution N(0,1). To convert this to any normal distribution N(mu, sigma^2) => sigma * N(0,1) + mu. For mu=0 and var=sigma^2=2D dt, we indeed need to change the multiplicative factor in the free (Brownian) equation from sqrt(D dt) to sqrt(2 D dt). This modification was done in both the software and in the paper.

Also, in the context of the movie simulator, we used the SNR parameter as a way to adjust the contrast between the simulated particle intensity and the background intensity (contrast = particle intensity – background intensity). To better represent this, we have renamed this parameter to “contrast” in the GUI, the code, the documentation, and the paper.

Trait2d_analysis gui_gui:

- The warnings produced by the analysis GUI were the result of an unchecked bug. This has been fixed and no warnings should be produced now. We would like to thank the reviewer for bringing this to our attention.

- The “Max. fit iterations” parameter is internally passed along to the “maxfev” (maximum function evaluations) parameter of scipy.optimize.curve_fit, which is responsible for performing the actual model fits. In case fits fail, it might help to increase this parameter.

- If the “Use parameters as initial guesses” checkbox is checked, the output parameters of the last fit will be passed to the “p0” parameter of scipy.optimize.curve_fit and used as initial guesses for the next fit. Similar to “Max. fit iterations” this might be used to improve existing fits in some scenarios.

- The Range parameter is, indeed, equivalent to the fix_max_time argument present in the code at various points. The issue of proper range determination is already presented in the paper. A link to the paper was added to the README file in the repository, so that it can serve as further documentation for the user, together with the present reviewer report. Once again, we extend an invitation to the reviewer to write an issue on the GitHub repository on this topic, following the finalization of the review process, to direct further developments of the tool.

Trait2d_simulator_gui:
Thank you for these suggestions. We modified the default simulation parameters to make sure that the “Free diffusion” and “Hopping Diffusion” models can be used without necessitating too many resources. We added a dialog box to inform the user of the generated movie size before performing the movie simulation. Also, information about the simulation process is shown in the terminal.

Trait2d_tracker_gui:

Taking into account the reviewer suggestions, we now have updated the tracker with the following features:

- Import 16-bit TIFF stack as well as 8-bit
- Save/load parameters of the tracker
- Test run of the linking algorithm, where the user can select the frame range to test the tracking before processing the entire image sequence. The detection results can be viewed directly in the viewer (preview button).

We have also provided an example stack with a parameter file, which can be found in the Github repository.

Typos

We thank the reviewer for pointing these out for our attention. We addressed and corrected all of them as advised.

References:

[1] M. Goulian and S. M. Simon, "Tracking Single Proteins within Cells," Biophys. J., vol. 79, no. 4, pp. 2188–2198, Oct. 2000, doi: 10.1016/S0006-3495(00)76467-8.
[2] G. Muñoz-Gil et al., "Objective comparison of methods to decode anomalous diffusion," Nat. Commun., vol. 12, no. 1, 2021, doi: 10.1038/s41467-021-26320-w.

Competing Interests: No competing interests were disclosed.