TrackNTrace

written for Matlab by Simon Christoph Stein and Jan Thiart

TrackNTrace is a fast, easy-to-use Matlab program created for locating bright, Gaussian-shaped spots in a 2D fluorescence microscopy image or movie and track such emitters over time. It can be used in localization microscopy (PALM, STORM, ...), Single-molecule Particle Tracking (SPT), for drift correction and similar applications and comes with a convenient interface for configuration. Locating and tracking fluorescent emitters is done in two separate steps and it is comparatively easy to everyone with basic Matlab knowledge to alter the respective scripts, e.g. to include additional tracking algorithms. TrackNTrace should be useful to anyone who seeks to combine the speed of established software packages like RapidSTORM or quickPALM with the simplicity and direct modifiablity of Matlab, especially when further post-processing is also done in Matlab.
In general, TrackNTrace reads a movie file, corrects for camera artifacts if applicable and obtains a rough guess of all possible positions of bright spots in every image. These position candidates then serve as the basis for a fitting routine which fits a 2D Gaussian bell curve to the pixel intensity values around each candidate as an approximation of the microscope Point Spread Function (PSF), obtaining position, amplitude and local background. Finally, these fit results are then returned to a particle tracking algorithm which tries to link particles close in time and space to form trajectories.

This manual will first provide all necessary steps for installation (section 1) and explain the configuration utility (section 2). Some in-depth information about the operation of TrackNTrace is provided in section 4.

Requirements

OS	Windows 7 64-bit or higher[1]
Matlab version	2009a or higher
Toolboxes	Image Processing, Statistics

Please also note: TrackNTrace currently can only handle single-channel, 2D+t Tif image stacks. Convert your experimental data accordingly.

[1] Necessary libraries can also be compiled for Linux and Mac.

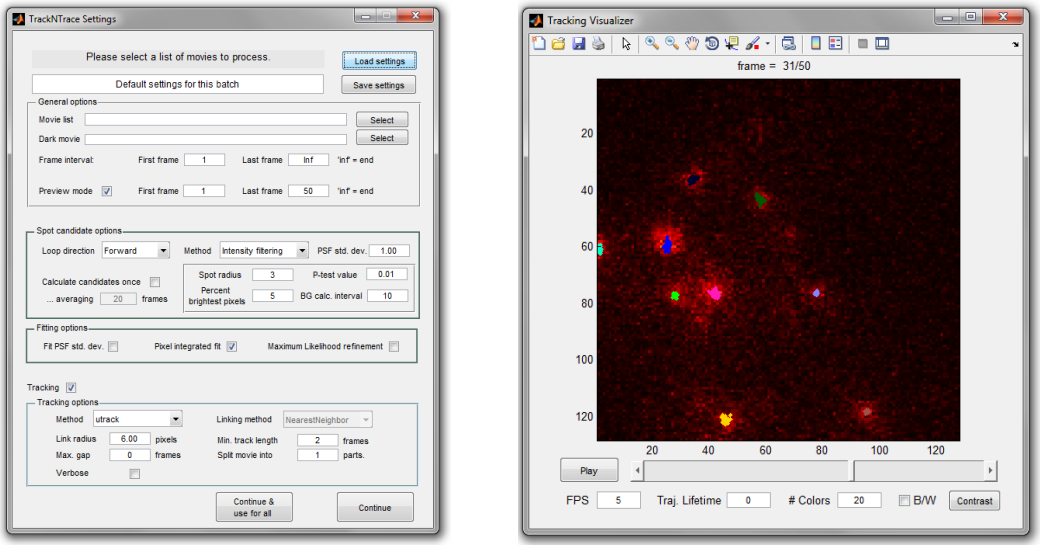
1 Installation

Installation procedure for Windows 7:

1. Extract `TrackNTrace.zip` to your Matlab root path directory (e.g. `C:\Users\%USERNAME%\Documents\MATLAB`).
2. Open the TrackNTrace folder and run `vcredist_x64.exe`.
3. Open Matlab and go to the TrackNTrace folder.
4. Run `RunTrackNTrace.m`.

2 Configuration

When you first run `RunTrackNTrace.m`, the GUI as seen in figure 1a pops up. Here, you can select the relevant movie files to process, set options to find spots and fit a PSF model to them, and handle the particle tracking algorithms. All relevant file names, options, particle locations, and trajectories are stored in a time-stamped `mat`-file whenever they become available which makes it very easy to repeat and compare different experiments and very hard to lose any progress. Before processing begins, users can enable a preview mode to try and optimize different settings, either individually for all selected movies or globally. Finally, finished movies can be inspected in Matlab with trajectories overlaid.



(a) GUI for TrackNTrace settings.

(b) GUI for preview mode.

Figure 1: TrackNTrace GUIs.

You can always save the current settings in a different `mat`-file and load them at the start by clicking on Save Settings and Load settings, respectively. In such a case, one only needs to select the movies to process and can skip anything else. Clicking on Continue & use for all will use the same settings for all selected movies while clicking Continue will invoke the GUI again for each individual movie.

General options

Movie list Click on Select to open a file dialogue where you can select one or multiple movies to analyze.

Dark movie Select a movie taken with the exact same camera settings used in your experiment but with the shutter closed. This movie is used to correct for non-isotropic camera sensitivity, dead pixels and other artifacts according to [1].

Frame interval Select the first and the last movie frame for processing. Anything not in this interval will be discarded. If Last frame is higher than the movie size or set to Inf, the whole movie is processed, starting at First frame.

Preview mode Enable previw mode and select its frame interval. If you click Continue and preview mode is enabled, all current settings will be applied to analyse a portion of the movie (keep the frame interval between 50 and 100 frames to avoid performance drops). A movie player (see fig. 1b) will pop up where you can see all positions and trajectories which are the result of the settings you choose. This is very useful for fine-tuning all relevant options to achieve the best result for your analysis.
Once you close the preview window, you will get asked to continue testing or abort the preview. Clicking Yes takes you back to the settings GUI, No acknowledges your satisfaction with the current settings and continues with the next film.

Spot candidate/Fitting options

Here, you can control how to roughly identify bright spots in your movies. These spot candidates are later fit and accurate positions and amplitudes get extracted.

Loop direction Movies can be processed in a forward direction, from beginning to end, or backwards. Normally, there is no difference between the two. However, this setting can be used in combination with Calculate candidates once (see below).

Calculate candidates once If enabled, spots are only detected once from an average over the first (or last, with Loop direction = Backward) N frames of the movie. The number of frames N can be adjusted below the checkbox. Instead of independent spot detection & fitting for every frame, the fit results of the last frame are carried over as an initial guess for the next frame. Useful for drift correction or blinking/bleaching analysis of non-moving particles at low signal to noise.

Method Choose the method used to find spot candidates, which can be intensity filtering or cross-correlation. The former works well in any case while the latter is very useful if looking at non-moving diffraction limited emitters.

PSF std. dev. TrackNTrace assumes an isotropic Gaussian PSF model for fluorescent emitters [2], where the provided value is the Gaussian standard deviation σ in pixels. A lower bound can be calculated as $\sigma = \frac{\lambda_{\mathrm{em}}}{4\sqrt{2\ln 2}\mathrm{NA}}$, where λ_{em} is the fluorescence emission wavelength in pixels and NA is the numerical aperture of the microscope. This number is very important when using cross-correlation.

Spot radius Average radius of a spot in pixels, must be an integer.

Percent brightest pixels Only the brightest $q\%$ of all pixels can be considerred as candidates. Lower means less candidates but higher candidate amplitude.

P-test value Every emitter's brightness is compared to its background via a hypothesis test, with the background assumed to be Gaussian. Lower means less candidates but higher candidate amplitude. Must be between 1 and 0.

BG calc. interval This setting forces a new local background image to be calculated every n frames. Setting a low number results in higher quality localizations at the cost of a severely reduced performance.

Threshold This number t is only used in the cross-correlation step and must be between 1 and 0. Choosing a large value for t will make sure only spots which strongly match the Gaussian PSF model are chosen as spot candidates.

Fit PSF std. dev. This enables fitting the PSF standard deviation on top of position, amplitude and background. Useful for wide particle size distributions or diffusing molecules.

Pixel integrated fit Enables or disables fitting a Gaussian model integrated over one pixel. It more accurately resembles the process of photons hitting the discrete pixel grid of a camera chip surface and is recommended to be enabled [2].

Maximum Likelihood refinement If enabled, the fitting algorithm uses a combination of Least-quares Minimization and Maximum Likelihood Estimation [2] for parameter optimization.

Tracking options

This section contains all settings relevant to particle tracking. Tracking can also be disabled altogether by unticking Tracking.

Method Choose the tracking algorithm you would like to use. Possible methods at this point are simpletracker [3], u-track [4], and IDLTracker [5].

Linking Method Only applies to simpletracker, determines greediness of algorithm. Nearest neighbor linking is more efficient but less accurate than Hungarian linking.

Link radius Maximum allowed distance a particle can travel between two frames. A good setting for diffusing particles is $3\sqrt{2Dt_a}$, where D is the diffusion coefficient and t_a is the frame aquisition time.

Max. gap Maximum amount of frames a particle can be invisible (e.g. by entering a non-radiative triplet state) between two observations without ending the trajectory prematurely. While useful for stationary, blinking fluorophores, allowing gaps for diffusing particles can severely distort the result. Set to 0 to disable.

Min. track length All trajectories shorter than the minimum allowed track length will be discarded automatically.

Split movie into parts: Particle tracking can be very memory-intensive and taxing on the CPU when dealing with a large amount of frames and particles, especially when gap closing is enabled. As performance does not scale linearly, splitting the movie into 2 or more parts can boost performance considerably. Choose with care if you do not want to allow tracks being split into smaller segments, e.g. when large gap closing values were chosen.

Verbose: If enabled, progress updates will be displayed if the tracking algorithm supports it.

3 TrackNTrace data structures

All results obtained by TrackNTrace are saved in a `mat`-file in the format `'moviename_timestamp_TNT.mat'`. Here is a list of all variables it contains:

TrackNTrace data structures

Variable name	Description
`filename_movie`	String containing full path of movie file.
`dark_img`	2D double array containing correction image to be added to each movie frame.
`generalOptions`	1×1 struct of general options variables set by the GUI.
`candidateOptions`	1×1 struct of candidate options variables set by the GUI.
`fittingOptions`	1×1 struct of fitting options variables set by the GUI.
`trackingOptions`	1×1 struct of general options variables set by the GUI.
`fitData`	$p \times k \times n$ double array of fitted positions where p is the number of model parameters, k is the maximum amount of particles in any frame, and n is the number of analyzed frames. Each column represents a unique fit and the row order according to the model PSF is μ_x, μ_y, A, B, σ and ε (see sec 4 for details). ε is a flag set by the fitting routine indicating if a fit converged ($\varepsilon = 1$) or not ($\varepsilon = -1$), or if the fit failed altogether ($\varepsilon = -2$). For example, `fitData(1:3,fitData(end,:)==1,f).'` shows a list of all valid particles in frame f in row order with xy position and amplitude as columns.
`trajectoryData`	2D double array of all trajectories. Each trajectory is assigned a unique id after which the array is sorted. The column order is id, frame, μ_x, μ_y, A. Both id and frame start at 1.

4 How TrackNTrace works

After storing all options, reading in the movie and, if applicable, a closed shutter movie for correction, the fitting routine starts. First, the movie frame is cleaned up by adding a correction image obtained from the closed shutter movie as described in [1]. Then, the respective candidate search function is called for the respective frame and the result is passed to the fit function. Depending on the settings, candidate search is

carried out in each frame or only once for the first (or last) frame or an average image of the first few (or last few) frames. In the latter case, fit results from the frame before are passed to the fit function in the next iteration.
Two candidate search mechanism are possible: Normalized cross correlation and intensity filtering. For cross correlation, a Gaussian PSF mask is created and the `normxcorr2` function is called to calculate a correlation image of the original movie frame. This image is then processed with a local maximum filter using `imdilate`. Pixels with a correlation value higher than the user-provided threshold are accepted as spot candidates. In the intensity filtering step, the image is convolved with a normalized filter kernel consisting of a moving average window and a Gaussian bell curve of $\sigma = 1$ px against discretization noise before being processed with a local maximum filter. The kernel has a window size of $w = 2r+1$ where r is the spot radius given by the user. Next, a local background image [4] is calculated and all pixels have to pass two tests: The intensity must be among the top $q\%$ in the image and $1 - \mathrm{CDF}(\mu_b, \sigma_b) \leq p$ must hold, where CDF ist the normal distribution cumulative density function and μ_b and σ_b are mean and standard deviation of the local background. p and q are user-provided thresholds.

A list of spot candidate positions is passed to a `Mex` file which handles all fitting procedures. First, a square quadrant I_{exp} around a candidate pixel is fit to a 2D Gaussian,

$$I_{\mathrm{theo}}(x,y) = A \exp\left(-\frac{(x-\mu_x)^2}{2\sigma^2} - \frac{(y-\mu_y)^2}{2\sigma^2}\right) + B \quad \text{with} \quad \min_\theta \sum_x \sum_y (I_{\mathrm{theo}} - I_{\mathrm{exp}})^2$$

where $\theta = (A, \mu_x, \mu_y, B, \sigma)$. Fitting σ can be disabled by the user. A more accurate model can be obtained by taking into account the finite size of the camera chip pixel grid, where all signal photons hitting any point within a square pixel are accumulated:

$$\begin{aligned} I_{\mathrm{theo,px}} &= \int_{-1/2}^{+1/2} \mathrm{d}x \int_{-1/2}^{+1/2} \mathrm{d}y \, I_{\mathrm{theo}} \\ &= \frac{A\pi\sigma^2}{2} \cdot \left[\mathrm{erfc}\left(-\frac{x-\mu_x+1/2}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(-\frac{x-\mu_x-1/2}{\sqrt{2}\sigma}\right)\right] \times \\ &\quad \left[\mathrm{erfc}\left(-\frac{y-\mu_y+1/2}{\sqrt{2}\sigma}\right) - \mathrm{erfc}\left(-\frac{y-\mu_x-1/2}{\sqrt{2}\sigma}\right)\right] + B \end{aligned}$$

Here, erfc is the complementary error function. Note that x- and y- dimension are decoupled, meaning that the two terms only have to be calculated once for one column and one line. Residual minimization is done by a Least-squares Levenberg-Marquardt algorithm. The result can be fitted again using Maximum Likelihood estimation which is proven to yield the best possible result [2]:

$$\min_\theta(-\log \mathcal{L}) = \min_\theta(I_{\mathrm{theo}}(\theta) - I_{\mathrm{exp}} \ln I_{\mathrm{theo}}(\theta))$$

All optimization steps rely on the ceres-solver library, an open-source optimization library written in C++ by Google Inc. We chose ceres for its very high performance, great customizability and the possibility of calculating all necessary first-order derivatives by using automatic differentiation which does not require

user input and is therefore more robust. The matlab implementation by Simon Christoph Stein is available on the Matlab file exchange where you can also find all necessary files and instructions for building TrackNTrace: `http://www.mathworks.com/matlabcentral/fileexchange/52417-fast-gaussian-point-spread-function-fitting--mex-`

After fitting is finished and if tracking is enabled, the result array is converted to a format suitable for the chosen tracker and passed to the tracking routine. It is highly recommended to read their respective publications and instructions [3–5]. Depending on the tracker, the time complexity of particle linking is at least quadratic in the number of particles, linear in the number of frames and quadratic or even exponential in the number of gap frames. In some cases, splitting the movie into several parts can be much faster than processing all positions in a single batch. In such cases, particles within and directly after such a border or split frame are tracked again to try to re-link trajectory segments artificially cut in half by splitting the movie. Gap frames cannot be accounted for in this case, therefore, caution is advised when using larger gap values and splitting numbers together.
For every tracker, the end result is an array containing all trajectories with recorded frame number, and respective xy- position and amplitude. Every trajectory has a unique id to facilitate post-processing.
While all tracking algorithms are handled more or less the same, u-track is an exception. u-track can handle very difficult scenarios such as Brownian-directional motion-switchting, particle merging and splitting and provides a large number of user-input variables for this. These options are hidden in the `parseUtrackOptions.m` file and are only meant to be changed by an experienced user. Caution is advised.

5 References

[1] Hirsch M, Wareham RJ, Martin-Fernandez ML, Hobson MP, Rolfe DJ: A Stochastic Model for Electron Multiplication Charge-Coupled Devices – From Theory to Practice. PLoS ONE 8(1), 2012.

[2] Mortensen, KI, Churchman SL, Spudich, JA, Flyvbjerg H: Optimized localization analysis for single-molecule tracking and super-resolution microscopy. Nature Methods 7, 377 - 381, 2010.

[3] Tinevez, Jean-Yves: Simple Tracker. Matlab File Exchange, `http://de.mathworks.com/matlabcentral/fileexchange/34040-simple-tracker` retrieved 2015-08-13.

[4] Jaqaman K et al: Robust single-particle tracking in live-cell time-lapse sequences. Nature Methods 5, 695 - 702, 2008.

[5] Crocker JC, Grier DG: Methods of Digital Video Microscopy for Colloidal Studies. J. Colloid Interface Sci. 179, 298, 1996.