README for running the Matlab implementation of the tracking algorithm described in the paper “Robust single particle tracking in live cell time-lapse sequences” by **Jaqaman et al., Nature Methods 5: 695-702 (2008)**.

TABLE OF CONTENTS:

Installation:

The download contains **two code directories (u-track and winExtra)** and **one example dataset directory**.

Before running the software, add the **u-track** directory to the Matlab path (File -> Set Path -> Add with Subfolders). Do not add **winExtra** (see below).

The software requires the following **Matlab toolboxes**:
- Image Processing
- Optimization
- Statistics
- Curve Fitting

The software has been tested on **Matlab R2011a** under **Linux 64-bit, Windows 64-bit** and **Mac OSX 10.6.8**. In the past, the software was also tested on Linux 32-bit and Windows 32-bit, so most probably it will run on these platforms, but these platforms are no longer explicitly supported.

The software contains two functions written in C/C++ (*createDistanceMatrix* and *mexLap*), that have been compiled for the platforms listed above. The source codes are supplied in case the functions need to be compiled for different platforms.

Note: Windows XP and Windows 7 on 64-bit machines require different executables. The default supplied Windows 64-bit executables are for Windows 7. The executables for Windows XP 64-bit can be found in “winExtra.” To run the software on a Windows XP 64-bit machine, please swap createDistanceMatrix.mexw64 and mexLap.mexw64 between winExtra and u-track.

As in the paper, in our terminology **tracking** is distinct from **detection**. **Detection** locates the particles in each frame of a movie, while **tracking** uses the detection results and links the detected particles into trajectories.

Running the software:

You can run u-track either from the command line or via a GUI, as described below.

In either case, each movie to be analyzed must be stored as tiff files, with one tiff file per time point and the tiff files named filename_0001.tiff, filename_0002.tiff, filename_0003.tif, etc. Please start with the number 1, not 0. The index should always have the same number of digits for a series of tiff images (e.g. images 1, 10 and 100 of a movie will have the indices 0001, 0010, 0100).

GUI:

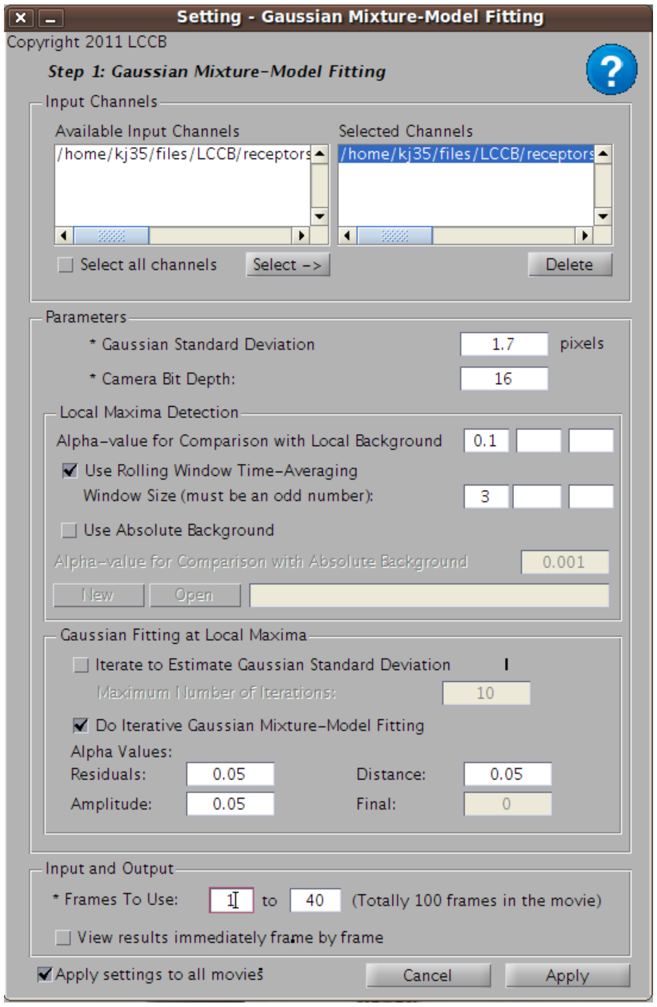
In the matlab command line, type

```
>> uTrackPackageGUI
```

and follow the instructions. There are help buttons with attached help files for guidance.

Below are parameters that work well for the supplied example dataset:

Detection – Gaussian mixture-model fitting:


Setting - Gaussian Mixture-Model Fitting
Copyright 2011 LCCB
Step 1: Gaussian Mixture-Model Fitting

Input Channels
Available Input Channels: /home/kj35/files/LCCB/receptors
Selected Channels: /home/kj35/files/LCCB/receptors
☐ Select all channels

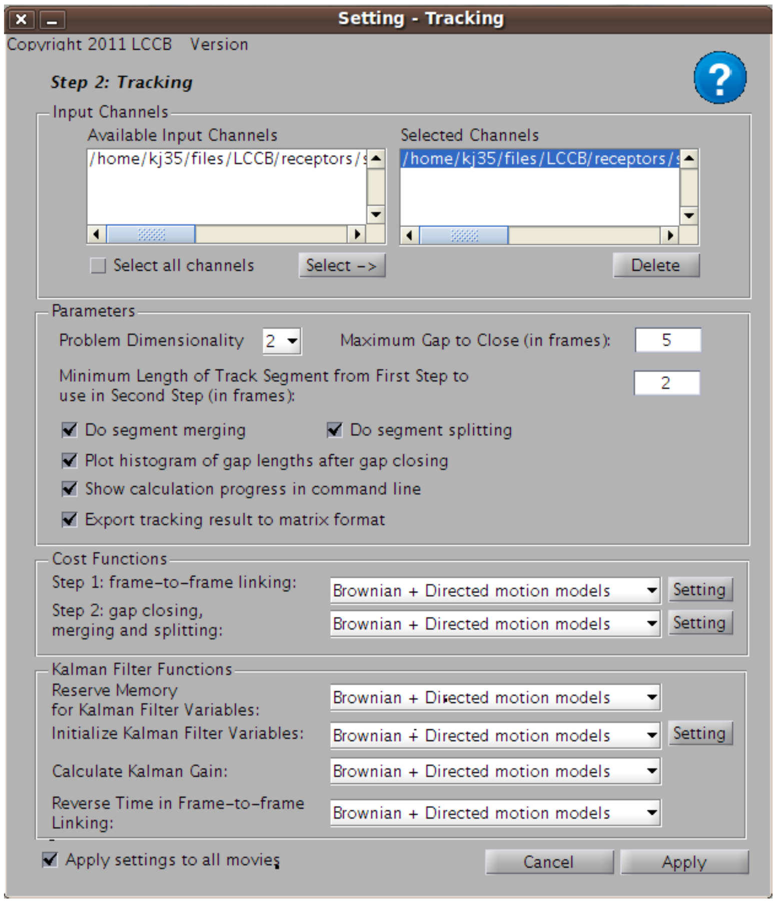
Parameters
* Gaussian Standard Deviation: 1.7 pixels
* Camera Bit Depth: 16

Local Maxima Detection
Alpha-value for Comparison with Local Background: 0.1
☑ Use Rolling Window Time-Averaging
Window Size (must be an odd number): 3
☐ Use Absolute Background
Alpha-value for Comparison with Absolute Background: 0.001

Gaussian Fitting at Local Maxima
☐ Iterate to Estimate Gaussian Standard Deviation
Maximum Number of Iterations: 10
☑ Do Iterative Gaussian Mixture-Model Fitting
Alpha Values:
Residuals: 0.05 Distance: 0.05
Amplitude: 0.05 Final: 0

Input and Output
* Frames To Use: 1 to 40 (Totally 100 frames in the movie)
☐ View results immediately frame by frame
☑ Apply settings to all movies

Tracking – general setup:



Input Channels

Available Input Channels: /home/kj35/files/LCCB/receptors/s

Selected Channels: /home/kj35/files/LCCB/receptors/s

☐ Select all channels   Select ->   Delete

Parameters

Problem Dimensionality: 2

Maximum Gap to Close (in frames): 5

Minimum Length of Track Segment from First Step to use in Second Step (in frames): 2

☑ Do segment merging   ☑ Do segment splitting

☑ Plot histogram of gap lengths after gap closing

☑ Show calculation progress in command line

☑ Export tracking result to matrix format

Cost Functions

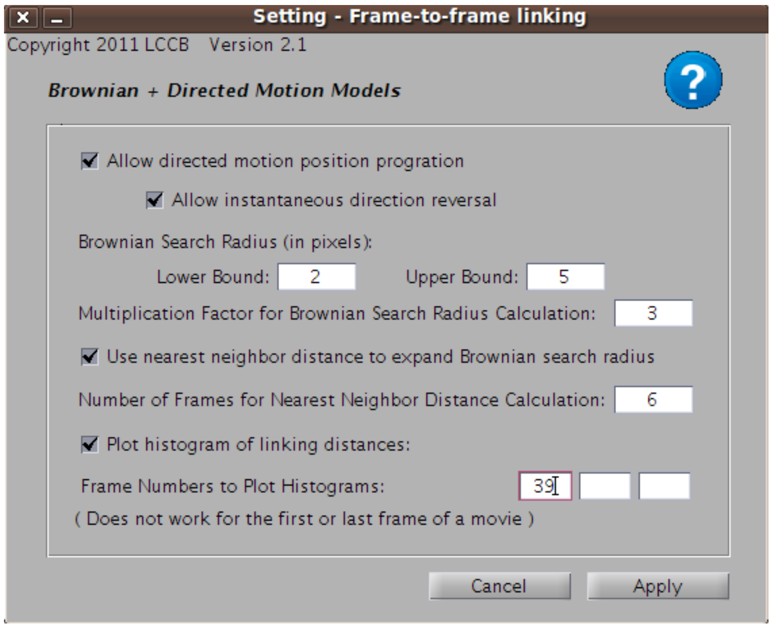
Step 1: frame-to-frame linking:	Brownian + Directed motion models	Setting
Step 2: gap closing, merging and splitting:	Brownian + Directed motion models	Setting

Kalman Filter Functions

Reserve Memory for Kalman Filter Variables:	Brownian + Directed motion models	
Initialize Kalman Filter Variables:	Brownian + Directed motion models	Setting
Calculate Kalman Gain:	Brownian + Directed motion models	
Reverse Time in Frame-to-frame Linking:	Brownian + Directed motion models	

☑ Apply settings to all movies   Cancel   Apply

Tracking – Cost function for frame-to-frame linking:


Setting - Frame-to-frame linking
Copyright 2011 LCCB Version 2.1
Brownian + Directed Motion Models
?

☑ Allow directed motion position progration

☑ Allow instantaneous direction reversal

Brownian Search Radius (in pixels):

Lower Bound: 2   Upper Bound: 5

Multiplication Factor for Brownian Search Radius Calculation: 3

☑ Use nearest neighbor distance to expand Brownian search radius

Number of Frames for Nearest Neighbor Distance Calculation: 6

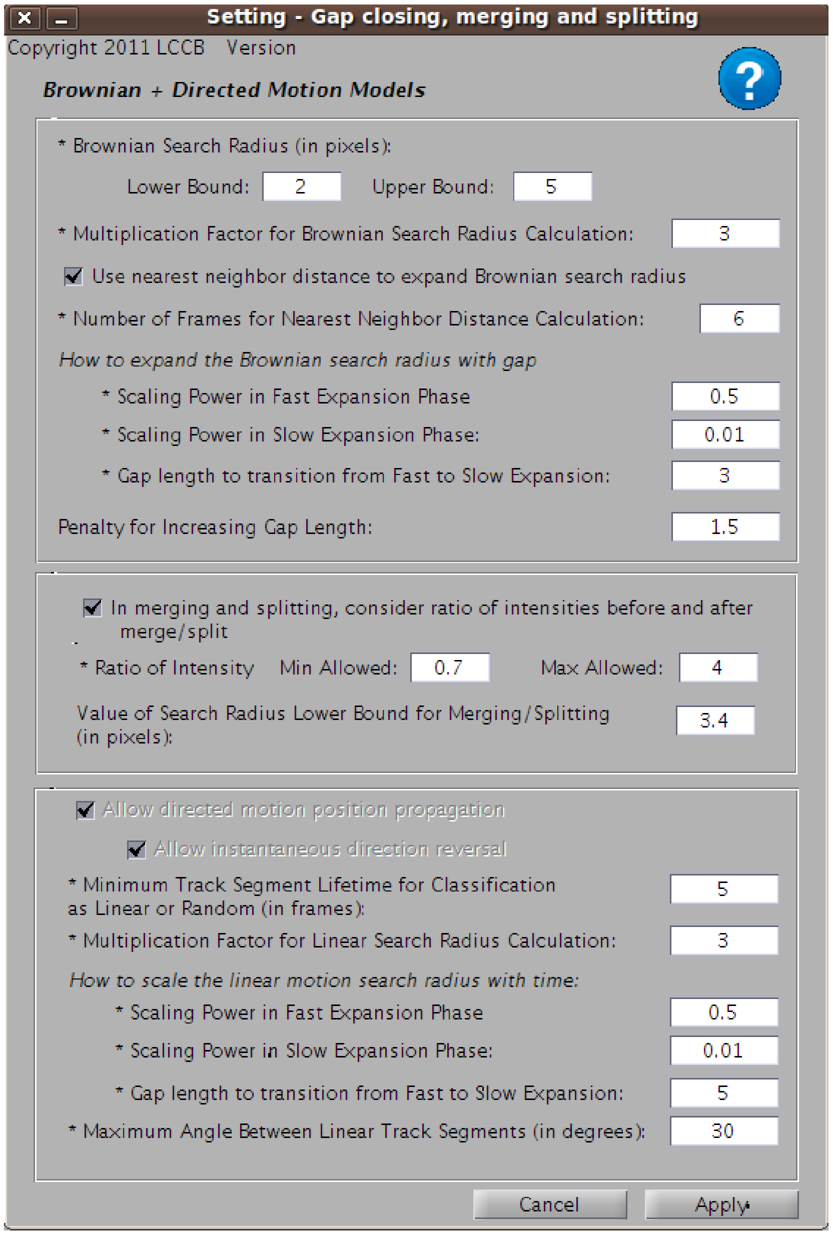
☑ Plot histogram of linking distances:

Frame Numbers to Plot Histograms: 39

(Does not work for the first or last frame of a movie)

Cancel   Apply

Tracking – Cost function for gap closing, merging and splitting:


Setting - Gap closing, merging and splitting

Copyright 2011 LCCB Version

Brownian + Directed Motion Models

* Brownian Search Radius (in pixels):

Lower Bound: 2 Upper Bound: 5

* Multiplication Factor for Brownian Search Radius Calculation: 3

☑ Use nearest neighbor distance to expand Brownian search radius

* Number of Frames for Nearest Neighbor Distance Calculation: 6

How to expand the Brownian search radius with gap

* Scaling Power in Fast Expansion Phase 0.5
* Scaling Power in Slow Expansion Phase: 0.01
* Gap length to transition from Fast to Slow Expansion: 3

Penalty for Increasing Gap Length: 1.5

☑ In merging and splitting, consider ratio of intensities before and after merge/split

* Ratio of Intensity Min Allowed: 0.7 Max Allowed: 4

Value of Search Radius Lower Bound for Merging/Splitting (in pixels): 3.4

☑ Allow directed motion position propagation

☑ Allow instantaneous direction reversal

* Minimum Track Segment Lifetime for Classification as Linear or Random (in frames): 5
* Multiplication Factor for Linear Search Radius Calculation: 3

How to scale the linear motion search radius with time:

* Scaling Power in Fast Expansion Phase 0.5
* Scaling Power in Slow Expansion Phase: 0.01
* Gap length to transition from Fast to Slow Expansion: 5
* Maximum Angle Between Linear Track Segments (in degrees): 30

NOTE: The analysis part of uTrackPackageGUI is not backward compatible. The visualization part might be.

<u>**Command line:**</u>

<u>Detection:</u>

We supply a code (**scriptDetectGeneral**) for the detection of diffraction-limited objects (such as single molecules and small molecular aggregates that are below the resolution limit) as presented in **Supplementary Note 2** of the paper. Note that this detection code is not optimal for detecting objects with variable size above the resolution limit.

Importantly, the detection step is application specific. Thus, the detection and tracking codes in the software package are not linked and **alternative detection codes can be used**.

The detection results must be presented in the following format (which is the automatic output of scriptDetectGeneral):

The **detection results** should be in a Matlab structure called **movieInfo** (Fig. 1). If the movie consists of **N frames**, movieInfo will be an **N×1 structure array** (i.e. containing one entry per frame). The structure movieInfo must contain at least **3 fields**: **xCoord** (x-coordinates of particles), **yCoord** (y-coordinates of particles) and **amp** (intensities of particles). If the application is 3D, then there must be a **4th field**, **zCoord** (z-coordinates of particles). If there are **P particles in frame i**, each of xCoord, yCoord, zCoord (if applicable) and amp will be a **P×2 array** (i.e. one row per particle). The first column contains the values (e.g. x-coordinates of particles). The second column defines the values' standard deviations (if unknown, put 0).

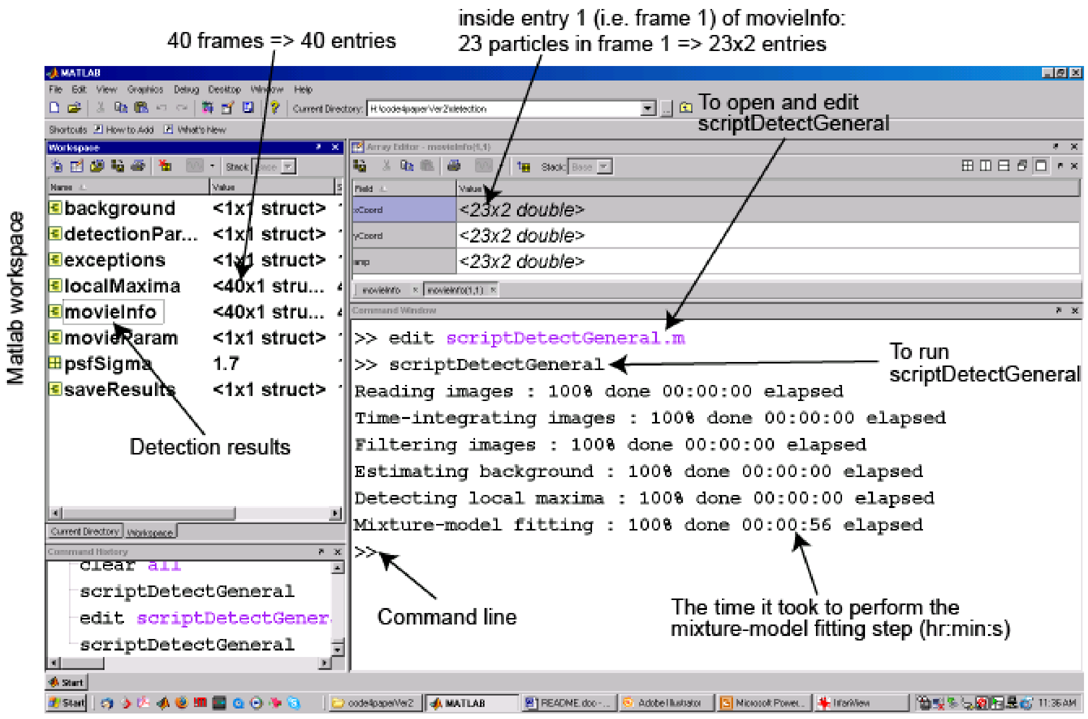
To run the detection code:

(1) Open the file scriptDetectGeneral.m from the Matlab command line (Fig. 1).
(2) Define the required parameters (explained in scriptDetectGeneral):
- (i) Movie parameters.
- (ii) Detection parameters.
- (iii) Saving results: Indicate the name and location of the file where results should be saved.

(3) Save the changes.
(4) Run scriptDetectGeneral from the Matlab command line (Fig. 1).

The parameters supplied in scriptDetectGeneral are suitable for the example dataset. However, the paths of the images and the results must be adjusted to the specific configuration of the software installation.

The output of scriptDetectGeneral, the variable movieInfo, will be saved in the file specified in Point (2.iii) above, and it will appear in the Matlab workspace for use in the tracking code (Fig. 1).


40 frames => 40 entries
inside entry 1 (i.e. frame 1) of movieInfo:
23 particles in frame 1 => 23x2 entries
To open and edit scriptDetectGeneral
Matlab workspace
To run scriptDetectGeneral
Detection results
Command line
The time it took to perform the mixture-model fitting step (hr:min:s)

Name	Value
background	<1x1 struct>
detectionPar...	<1x1 struct>
exceptions	<1x1 struct>
localMaxima	<40x1 stru...
movieInfo	<40x1 stru...
movieParam	<1x1 struct>
psfSigma	1.7
saveResults	<1x1 struct>

Field	Value
xCoord	<23x2 double>
yCoord	<23x2 double>
amp	<23x2 double>

```
>> edit scriptDetectGeneral.m
>> scriptDetectGeneral
Reading images : 100% done 00:00:00 elapsed
Time-integrating images : 100% done 00:00:00 elapsed
Filtering images : 100% done 00:00:00 elapsed
Estimating background : 100% done 00:00:00 elapsed
Detecting local maxima : 100% done 00:00:00 elapsed
Mixture-model fitting : 100% done 00:00:56 elapsed
>>
```

```
clear all
scriptDetectGeneral
edit scriptDetectGener
scriptDetectGeneral
```

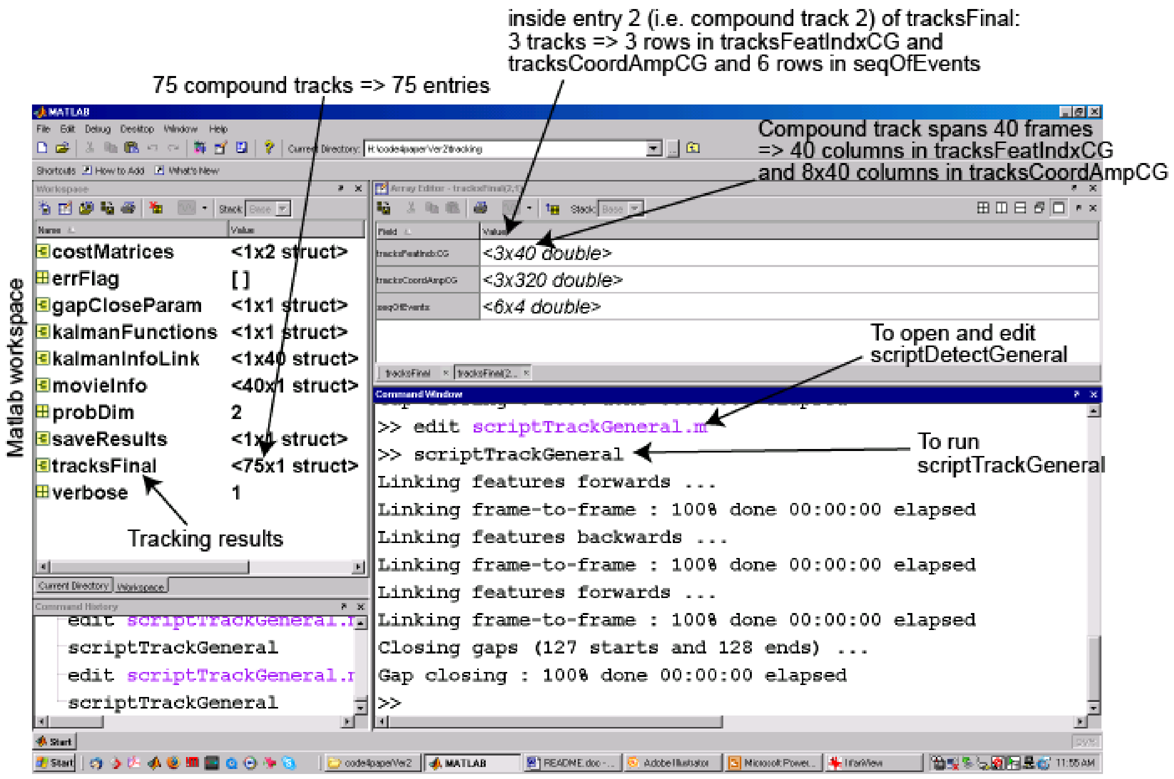
Figure 1: Illustration of a successful detection run and output for the example movie in the directory “example”. Note: The version of the detection code supplied in the current software package will give the user a different progress report, but algorithmically the detection runs in the same way.

Tracking:

After running the detection code, and with the results **movieInfo in the Matlab workspace** (either directly from scriptDetectGeneral of by loading the file where the detection results were saved into the Matlab workspace), **run the tracking code scriptTrackGeneral:**

(1) Open the file scriptTrackGeneral.m from the Matlab command line (Fig. 2).
(2) Define the required parameters (explained in scriptTrackGeneral, in **Supplementary Note 9** and in the accompanying document **SoftwareModifications111219**):
  (i) Names of cost functions and Kalman Filter functions.
  (ii) General and cost function-specific tracking parameters.
  (iii) Saving results: Indicate the name and location of the file where results should be saved.
(3) Save the changes.
(4) Run scriptTrackGeneral from the Matlab command line (Fig. 2).

The parameters supplied in scriptTrackGeneral are suitable for the example dataset. However, the path of the results must be adjusted to the specific configuration of the software installation.


inside entry 2 (i.e. compound track 2) of tracksFinal:
3 tracks => 3 rows in tracksFeatIndxCG and
tracksCoordAmpCG and 6 rows in seqOfEvents
75 compound tracks => 75 entries
Compound track spans 40 frames
=> 40 columns in tracksFeatIndxCG
and 8x40 columns in tracksCoordAmpCG
Matlab workspace
To open and edit
scriptDetectGeneral
To run
scriptTrackGeneral
Tracking results

Name	Value
costMatrices	<1x2 struct>
errFlag	[]
gapCloseParam	<1x1 struct>
kalmanFunctions	<1x1 struct>
kalmanInfoLink	<1x40 struct>
movieInfo	<40x1 struct>
probDim	2
saveResults	<1x1 struct>
tracksFinal	<75x1 struct>
verbose	1

Field	Value
tracksFeatIndxCG	<3x40 double>
tracksCoordAmpCG	<3x320 double>
seqOfEvents	<6x4 double>

```
>> edit scriptTrackGeneral.m
>> scriptTrackGeneral
Linking features forwards ...
Linking frame-to-frame : 100% done 00:00:00 elapsed
Linking features backwards ...
Linking frame-to-frame : 100% done 00:00:00 elapsed
Linking features forwards ...
Linking frame-to-frame : 100% done 00:00:00 elapsed
Closing gaps (127 starts and 128 ends) ...
Gap closing : 100% done 00:00:00 elapsed
>>
```

```
edit scriptTrackGeneral.m
scriptTrackGeneral
edit scriptTrackGeneral.m
scriptTrackGeneral
```

Figure 2: Illustration of a successful tracking run and output for the example movie in the directory “example”.

The important output of scriptTrackGeneral is the **output variable tracksFinal** (Fig. 2). If the tracker finds **N tracks** in the movie, tracksFinal will be an **N×1 structure array** (i.e. every element corresponds to one track). These tracks are **compound tracks**. If, for example, two particles first move separately and then they merge, their tracks will appear together as one compound track entry in tracksFinal.

Every entry in tracksFinal (i.e. every compound track) contains **3 fields**:

(1) tracksFeatIndxCG: Connectivity matrix of particles between frames, after gap closing.
Number of rows = Number of tracks merging with each other and splitting from each other (i.e. involved in compound track).
Number of columns = Number of frames the compound track spans.
Zeros indicate frames where particles do not exist, either because those frames are before the track starts or after it ends, or because of temporary particle disappearance.

(2) tracksCoordAmpCG: The positions and amplitudes of the tracked particles, after gap closing.
Number of rows = Number of tracks merging with each other and splitting from each other (i.e. involved in compound track).
Number of columns = 8 × number of frames the compound track spans. For every frame, the matrix stores the particle’s x-coordinate, y-coordinate, z-coordinate (0 if 2D),

amplitude, x-coordinate standard deviation, y-coordinate standard deviation, z-coordinate standard deviation (0 if 2D) and amplitude standard deviation.

(3) seqOfEvents: Matrix storing the sequence of events in a compound track (i.e. track start, track end, track splitting and track merging).
Number of rows = number of events in a compound track (= 2 x number of tracks within the compound track).
Number of columns = 4.
In every row, the columns mean the following:
1st column indicates frame index where event happens.
2nd column indicates whether the event is the start or end of a track. 1 = start, 2 = end.
3rd column indicates the index of the track that starts or ends (The index is “local”, within the compound track. It corresponds to the track’s row number in tracksFeatIndxCG and tracksCoordAmpCG).
4th column indicates whether a start is a true initiation or a split, and whether an end is a true termination or a merge. If the 4th column is NaN, then a start is an initiation and an end is a termination. If the 4th column is a number, then the start is a split and the end is a merge, where the track of interest splits from / merges with the track indicated by the number in the 4th column.

The output tracksFinal will be saved in the file specified in Point (2.iii) above, and it will appear in the Matlab workspace for use in the visualization code (Fig. 2).

We also supply two functions that allow the conversion of the tracks from this structure format into a matrix format that might be easier for further track processing.

CAUTION: The matrix format can take a lot of memory, so be careful when using this conversion for large movies. Also, merging and splitting information will be lost with this conversion.

If tracking was done WITHOUT merging and splitting, call the command

```
>> [trackedFeatureInfo, trackedFeatureIndx] =
convStruct2MatNoMS(tracksFinal);
```

If tracking was done WITH merging and/or splitting, call the command:

```
>> [trackedFeatureInfo, trackedFeatureIndx, trackStartRow, numSegments]
= convStruct2MatIgnoreMS(tracksFinal);
```

Input:
-tracksFinal: This is the output of scriptTrackGeneral. This variable should exist in the Matlab workspace after running scriptTrackGeneral.

Output:
-trackedFeatureInfo: This is the equivalent of tracksCoordAmpCG above, but for all the

tracks together.
-trackedFeatureIndx: This is the equivalent of tracksFeatIndxCG above, but for all the tracks together.
-trackStartRow: An array indicating the row in the big matrices storing the information of the first segment of each compound track.
-numSegments: An array indicating the number of segments belonging to each compound track.

Visualization:

There are four visualization functions:

(1) plotTracks2D: Statically plots the tracks generated by scriptTrackGeneral. The function can be called from the Matlab command line as follows:

```
>> plotTracks2D(tracksFinal, timeRange, colorTime, [],  indicateSE,
newFigure, image, [], ask4sel, [], minLength)
```

It takes the following input:
-tracksFinal: This is the output of scriptTrackGeneral. This variable should exist in the Matlab workspace after running scriptTrackGeneral.
-timeRange: Frame range to be plotted, e.g. [20 40] plots the tracks that exist between frames 20 and 40. To plot tracks in the whole movie, enter [].
-colorTime:
-'1' to draw tracks with time color-coding (tracks start in green, go through blue, end in red). Warning: This option makes the figure very big and slow if there are many tracks.
-'2' to cycle through 7 colors and give every track a random color.
-'3' to cycle through 23 colors and give every track a random color.
-'r', 'b', 'g', etc. to plot all the tracks in the same color, which is the color indicated between quotes (r=red, b=blue, g=green, etc (in Matlab, type 'help plot' for line colors)).
-indicateSE: 1 to indicate track starts and ends with circles and squares, respectively; 0 not to.
-newFigure: 1 to plot tracks in a new figure window, 0 to plot them in an already open figure window (e.g. to overlay two sets of tracks).
-image: In order to overlay the tracks onto an image (e.g. the first image in a movie), supply the image (as a uint16 or double matrix which already exists in the Matlab workspace). In this case, newFigure has to be 1.
-ask4sel: 1 to select tracks and get their information, 0 otherwise. If 1, there will be a question via the command line about selecting tracks. Answer "y" for yes or "n" for no. If "y," then you can click on points in the plot and get back information about the tracks. Use left-click, until the last point where a right-click is needed. After you get the information about the requested tracks, you can repeat this process until no longer desired. YOUR FINAL CHOICE MUST BE "n," OTHERWISE MATLAB MIGHT

HANG UP.
-minLength: Minimum length (duration in frames) of a track to be included in plot.

If plotTracks2D is called with only the first input argument, i.e. one types the command plotTracks2D(tracksFinal) in the Matlab command line, the code's default is to plot all the tracks for the whole movie, with all the tracks colored black, showing track starts and ends, in a new figure window without an image underneath.

In the tracks, a dotted line = closed gap, a dashed line = merge, a dash-dotted line = split.

(1) plotCompTrack: Plots one track at a time, showing coordinates and amplitude vs. time. The function can be called from the Matlab command line as follows:

>> plotCompTrack(trackToBePlotted)

It takes the following input:
-trackToBePlotted: This is one of the tracks from the output of scriptTrackGeneral. For example, to plot track # 10, define trackToBePlotted = tracksFinal(10). This variable should exist in the Matlab workspace after running scriptTrackGeneral.

In the tracks, a dotted line = closed gap, a dashed line = merge, a dash-dotted line = split.

(3) overlayFeaturesMovie: Generates a Quicktime movie of the detected features overlaid on the analyzed images. The function can be called from the Matlab command line as follows:

>> overlayFeaturesMovie(movieInfo, startend, saveMovie, movieName, [], showRaw, intensityScale, firstImageFile, dir2saveMovie)

It takes the following input:
-movieInfo: This is the output of scriptDetectGeneral. This variable should exist in the Matlab workspace after running scriptDetectGeneral.
-startend: Frame range to be plotted, e.g. [20 40] plots the detected features between frames 20 and 40. To plot all frames, enter [].
-saveMovie: 1 to save movie with overlaid features as a Quicktime movie; 0 otherwise.
-movieName: Name of movie, for example "detectionResults.mov", if saving is requested. Enter [] if movie is not to be saved.
-showRaw: 1 to show original movie to the left of the overlaid movie, 2 to show original movie to the top of the overlaid movie, 0 to only show overlaid movie.
-intensityScale: 0 to autoscale every image in the movie, 1 to have a fixed scale using intensity mean and std, 2 to have a fixed scale using minimum and maximum intensities.
-firstImageFile: Full name (including path) of first image file for overlaying. The file has to be the first image that has been analyzed even if not plotted. If file is not specified, i.e. if [] is used, user will be prompted to select the first image.

-dir2saveMovie: Directory where to save output movie. If not input, i.e. if [] is used, movie will be saved in directory where images are located by default.

(4) overlayTracksMovieNew: Generates a movie of the tracks overlaid on the analyzed images. In the movie, detected objects will be indicated by circles, closed gaps by asterisks, merges by yellow diamonds and splits by green diamonds.

The function can be called from the Matlab command line as follows:

```
>> overlayTracksMovieNew(tracksFinal, startend, dragtailLength,
saveMovie, movieName, [], 0, highlightES, showRaw, imageRange, 0,
classifyLft, [], intensityScale, colorTracks, firstImageFile, dir2saveMovie,
minLength)
```

It takes the following input:
-tracksFinal: This is the output of scriptTrackGeneral. This variable should exist in the Matlab workspace after running scriptTrackGeneral.
-startend: Frame range to be plotted, e.g. [20 40] plots the detected features between frames 20 and 40. To plot all frames, enter [].
-dragtailLength: Length of track drag tail. To show tracks for the whole length of a movie, enter a value larger than the movie length.
-saveMovie: 1 to save movie with overlaid features as a Quicktime movie; 0 otherwise.
-movieName: Name of movie, for example "trackingResults.mov", if saving is requested. Enter [] if movie is not to be saved.
-highlightES: 1 to highlight object appearance and disappearance (appearance in green, disappearance in yellow); 0 otherwise.
-showRaw: 1 to show original movie to the left of the overlaid movie, 2 to show original movie to the top of the overlaid movie, 0 to only show overlaid movie.
-imageRange: Image region to make movie out of, in pixels.
-classifyLft: 1 to color-code objects based on their lifetime (white: objects that last throughout the whole movie, red: objects that appear AND disappear in the movie, purple: objects that appear OR disappear (but not both) in the movie); 0 otherwise.
-intensityScale: 0 to autoscale every image in the movie, 1 to have a fixed scale using intensity mean and std, 2 to have a fixed scale using minimum and maximum intensities.
-colorTracks: 1 to give the track dragtails different colors, instead of color-coding the object symbols; 0 otherwise.
-firstImageFile: Full name (including path) of first image file for overlaying. The file has to be the first image that has been analyzed even if not plotted. If file is not specified, i.e. if [] is used, user will be prompted to select the first image.
-dir2saveMovie: Directory where to save output movie. If not input, i.e. if [] is used, movie will be saved in directory where images are located by default.
-minLength: Minimum length (duration in frames) of a track to be included in movie.