# Instructions for using the 3D tracking DLL.

This document is meant to provide a brief introduction on how to use the 3D tracking DLL to either track single particles in a feedback loop or acquire a z stack with the same setup.

Please refer to the original publication for additional information:

Spille et al., Direct observation of mobility state transitions in RNA trajectories by sensitive single molecule feedback tracking, Nucleic Acids Research (2014), doi: 10.1093/nar/gku1194

## 1. Description of files

- *3DTackCOR.cpp*: C++ source code for the 3D tracking DLL.
- *template1.txt*, *template2.txt*: Template data in text format to be read by the DLL. Contains the pixel values of the normalized templates in consecutive row order from top left to bottom right corner.
- *param_3DtrackCOR.txt*, *param_3DtrackCORstack.txt*: Text files containing parameters for tracking and stack acquisition mode respectively. See section 3 for a description of the parameters.

## 2. Getting started

During a real-time tracking experiment, the DLL is called after an image frame has been acquired. The algorithm localizes a particle based on information contained in the accompanying text files and returns a double type value corresponding to a voltage to be passed on to a piezo stage controller.

The files *template1.txt*, *template2.txt, param_3DtrackCOR.txt* and *param_3DtrackCORstack.txt* need to be saved on the lab computer running the image acquisition. The path to each of the files needs to be entered manually in the source code of *3DTackCOR.cpp*.

```
745     const char *DIRparam;
746     DIRparam = ".\\tracking\\param_3DtrackCOR.txt";         // tracking parameter txt file
747     const char *DIRstack;
748     DIRstack = ".\\tracking\\param_3DtrackCORstack.txt";    // stack parameter txt file
749     const char *DIRtmpl1;
750     DIRtmpl1 = ".\\tracking\\template1.txt";                // template 1 txt file
751     const char *DIRtmpl2;
752     DIRtmpl2 = ".\\tracking\\template2.txt";                // template 2 txt file
753     const char *DIRout;
754     DIRout   = ".\\tracking\\out";                          // output directory
```

**Fig. 1: Setting directories.**

The file can then be compiled as a DLL using e.g. Microsoft Visual C++ Express. The resulting 3DTrackCOR.dll needs to be placed in the search path of the main image acquisition software.
The algorithm was developed in our lab as an extension to the commercially available program ImSpector (version 5.5 including TrackDev module, LaVision BioTec, Bielefeld, Germany). The piezo stage was addressed via an appropriate controller (P-611.ZS stage and E-605 controller, Physik Instrumente, Karlsruhe, Germany). A voltage of 0 – 10 V applied to the E-605 BNC drives the stage to a position of 0 – 100 µm.
To protect the controller, the DLL return value zvolt is limited to this range in the *3DTackCOR.cpp* source code:

```
1039  // final check for valid output voltage:
1040      ////////////////////////////////////////
1041      ////////     stay in bounds!     //////////
1042      ////////////////////////////////////////
1043      if( zvolt < 0  ) zvolt = 0;
1044      if( zvolt > 10 ) zvolt = 10;
1045
1046      return zvolt;
1047      ////////////////////////////////////////
```

**Fig. 2: Safety limits for output voltage.**

## 3. Description of the tracking parameters

The files and contain all relevant parameters for either 3D feedback tracking or stack acquisition of reference structures in the same coordinate system. The latter is useful for overlaying tracking data with the appropriate slice of the reference stack. The DLL reads the parameter '*value*' placed between inverted commas in consecutive order.

a) *param_3DtrackCOR.txt*

| | |
|---|---|
| mode | '0' for stack acquisition, '1' for feedback tracking |
| xhme, yhme | initial image coordinates for tracking [pxl] |
| zhme | initial stage voltage (z position) for tracking [V]; 1 V = 10 µm |
| CALoff, CALslp | calibration curve offset and slope |
| LIMdiff | allowed particle displacement along each axis [pxl] |
| LIMsrch | allowed distance from initial coordinates when searching for a particle to track |
| LIMtmplt | template size is (2*LIMtmplt + 1) pixels in x and y |
| LIMcorr | size of image ROI for which normalized covariance is calculated is (2*LIMcorr + 1) pixels |
| LIMvarmin, LIMvarmax | min and max tolerable value of normalized covariance for valid particle |
| wait | wait so many frames for a particle to reappear within LIMdiff of its previous localization before returning to initial coordinates for new search |
| smooth | Apply [1 2 1; 2 4 2; 1 2 1]/16 kernel to smooth data before determining signal level? 0 = no, 1 = yes. |
| LIMdzmax | maximum allowed piezo stage displacement between frames [V] |
| LIMimin, LIMimax | min and max peak count level above background for a valid particle |
| doCOM | Calculate center of mass for refined lateral localization? 0 = no, 1 = yes. |

b) *param_3DtrackCORstack.txt*

| | |
|---|---|
| dz | axial displacement between slice [nm] assuming 1V = 10µm |
| frames p. slice | number of frames to be acquired for each axial position |
| slices | number of slices in stack |
| offset | position of first slice [µm], assuming 1V = 10µm |
| direction | 1: increase or -1: decrease voltage during acquisition |

## 4. Effect of the parameters

a) Feedback tracking

The effect of the parameters on the tracking procedure is depicted in Fig. 3

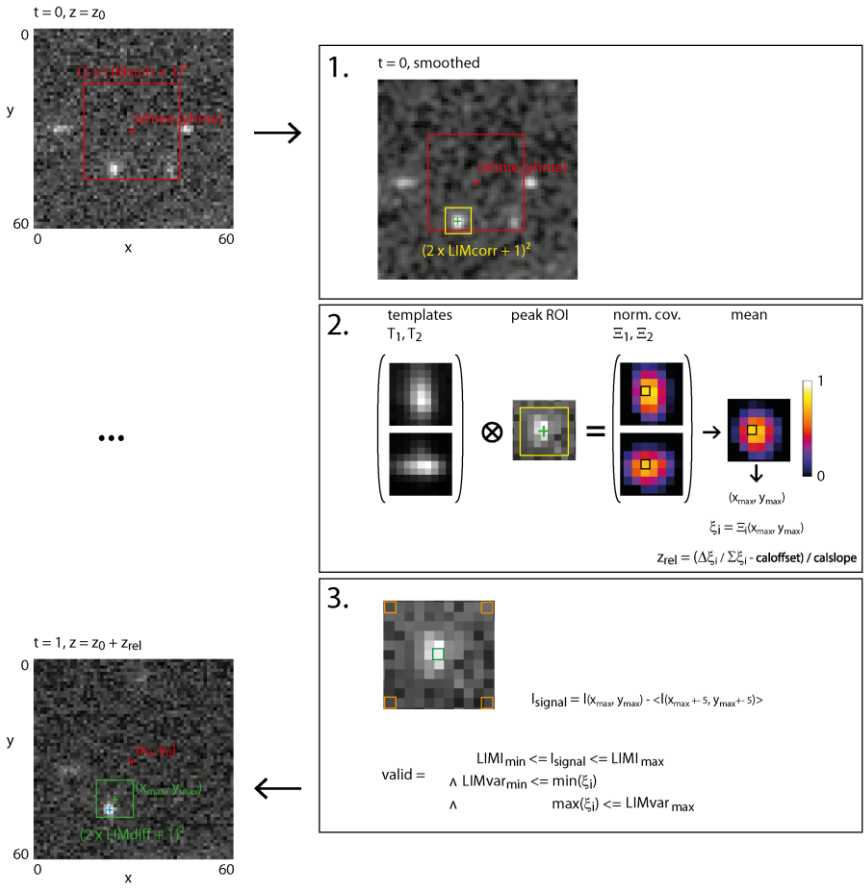



**Fig. 3: Tracking procedure.**

If mode = 1, feedback tracking is performed. Initially, the algorithm tries to locate a local intensity maximum within a distance *LIMsrch* along each image axis around the coordinates (*xhme*, *yhme*).

1. To facilitate the detection procedure, convolution with a [1 2 1; 2 4 2; 1 2 1]/16 kernel is applied. Optionally (*doCOM*), the center coordinate of the peak is refined by center of mass calculation.
2. For axial localization, the normalized covariance with each template T1, T2 is calculated. The calculation is performed with the raw data in a subimage of equal size as the templates. Obtained covariance values are assigned to the central pixel of the subimage, resulting in two normalized covariance images Ξ1, Ξ2. The pixel with the highest average of both values has the coordinates (*xmax*, *ymax*). For estimation of the relative axial position of the particle, the values ξ1 = Ξ (*xmax*, *ymax*) and ξ2 = Ξ2(*xmax*, *ymax*) are employed.
3. The background level is estimated as the average value of the four pixels separated +- 5 pixels along each axis from the peak coordinate. It is used to determine the signal level *Isignal* above background. If *Isignal* and the normalized covariance values stay within the boundaries set by *LIMimin*, *LIMimax*, *LIMvarmin*, *LIMvarmax*, the particle is considered *valid*

and an appropriate voltage *zvolt* = *z0* + *zrel* returned to the main program. The axial displacement between subsequent frames is additionally limited by *LIMdzmax* to 1V (corresponding to 10 µm). The coordinates (*xmax*, *ymax*) become the center for the particle search ROI in the subsequent image frame. As long as a valid particle was found in the previous frame, the search ROI dimensions are reduced to *LIMdiff*. To accommodate for brightness fluctuations or occasional large displacements, the algorithm accepts *wait* consecutive frames without valid localization in the search ROI before returning to the initial coordinates (*xhme*, *yhme*).

```
994             zrel = AxialLocalization(VARN1, VARN2, CALoff, CALslp);
995             if(zrel < -LIMdzmax) zrel = -LIMdzmax;                    // limit piezo displacement
996             if(zrel > LIMdzmax) zrel = LIMdzmax;
```

**Fig. 4: Determine axial displacement.**

b) Stack acquisition

Instead of determining the voltage value directed to the piezo stage controller from the image data, the algorithm can also return values determined from the parameters listed in . This functionality is usually used to acquire z stacks of reference structures in a different wavelength channel. Since voltages are registered both, during tracking and during stack acquisition, the appropriate reference images can readily be overlaid with the raw image data acquired during tracking.

The voltage to be issued to the piezo stage controller is determined from the offset corresponding to the position of the first slice, the number of frames to be acquired at each axial position, the direction of the stack and the separation between the slices as depicted in Fig. 5.

```
881 // calculate output voltage
882             slicecounter = counter / fperslice;         // determine slice number
883             framecounter = counter % nslice;            // frame within slice
884             zvolt = zoffset + dz * slicecounter;        // output voltage: offset + spacing * slice number
885             if( direction < 0){
886                 zvolt = zoffset - dz * slicecounter;    // ... but mind the sign
887             }
```

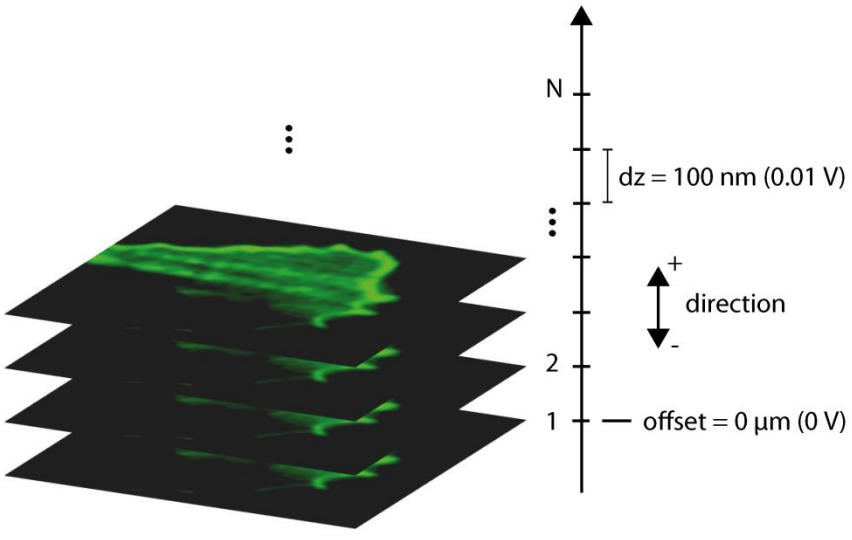



**Fig. 5: Stack acquisition.**

## 5. Data passed to and returned by the DLL

The parameters passed to the DLL are as follows:

```
12  ////////////////  DLL INPUT ////////////////////////////////////////////////////////
13  // unsigned short* image     - pointer to image data in PC memory
14  // int width, height         - width and height of image in pixels
15  // double** profiles         - pointer to pre-allocated array of size [profileLen x 8]
16  // float** timestamp         - pointer to pre-allocated array of size [profileLen x 8]
17  // int profileLen            - number of frames in image sequence; minimum is the number of pixels in a PSF template (see below)
18  // double adValue            - current piezo position [voltage]
19  // double daValue            - designated piezo position [voltage]
20  // unsigned int counter      - image counter; start with 0
21  // bool* imageChange         - modify image if changed during DLL execution?
22  ////////////////////////////////////////////////////////////////////////////////////
23
24  ////////////////  DLL OUTPUT ////////////////////////////////////////////////////////
25  // double zvolt              - voltage to be passed to piezo stage controller; here limited to 0 - 10V corresponding to 0 - 100 µm travel range
26  ////////////////////////////////////////////////////////////////////////////////////
```

```
62  // DLL entry point:
63  extern "C" __declspec(dllexport)double StartProcedure(unsigned short* image, int width, int height, double** profiles,
64      float** timestamp, int profileLen, double adValue, double daValue, unsigned int counter, bool* imageChange);
```

**Fig. 6: DLL input parameters and entry point.**

The only value returned to the caller is the double type variable *zvolt* corresponding to the new piezo stage goal voltage. The arrays profiles and timestamp are filled during an experiment and used to keep acquisition parameters in PC memory. They are occupied as depicted in Tab. 1. Values related to the tracking results are stored in *profiles* and the first three colums of *timestamp*.

| *profiles:* | | zcurr | zrel | znew | xpos | ypos | CC1 | CC2 | sgLevel |
|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| double | 0 | ... | ... | ... | ... | ... | ... | ... | ... |
| (8 bytes) | 1 | | | | | | | | |
| | . | | | | | | | | |
| | . | | | | | | | | |
| | . | | | | | | | | |
| | *N* | | | | | | | | ... |
| | | | | | | | | | |
| *timestamps:* | | time | valid | last | - | - | tmpl1 | tmpl2 | params |
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| float | 0 | ... | ... | ... | | | 0 | 0 | 0 |
| (4 bytes) | 1 | | | | | | ... | ... | tracking |
| | | | | | | | | | 22 |
| | . | | | | | | | | ... |
| | . | | | | | | | | 30 state |
| | | | | | | | | | 31 |
| | | | | | | | | | stack |
| | | | | | | | | | 36 |
| | | | | | | | ... | ... | ... |
| | | | | | | | *Ntmpl* | *Ntmpl* | |
| | *N* | | | | | | ... | ... | |

**Tab. 1: Use of arrays *profiles* and *timestamp*.**

Column 5 and 6 of timestamp are reserved for the template data from *template1.txt* and *template2.txt*, whereas column 8 of the array contains parameters loaded from *param_3DtrackCOR.txt* and *param_3DtrackCORstack.txt* as well as a status indicator for error messages in *timestamp*[7][30].

## 6. Output files

Data contained in the arrays as well as the tracking or stack acquisition parameters for the respective acquisition are saved to text files at the end of an experiment. The output folder is specified as shown in Fig. 1. For a tracking experiment, the tracking results text file contains a table with the following parameters (Tab. 2):

Number of frame within experiment – frame with last valid particle localization – timestamp for the frame – advalue [V] – devalue [V] – indicator of valid particle – lateral localization in x… – … and y – normalized covariance value with template 1… - … and template2 – signal count level above background

| frame# | last | time | advalue | davalue | valid | $x_c$ | $y_c$ | $\xi_1$ | $\xi_2$ | I($x_c$,$y_c$) |
|---|---|---|---|---|---|---|---|---|---|---|
| … | … | … | … | … | … | … | … | … | … | … |
| 12 | 9 | 282 | 4,9115 | 4,9166 | 0 | 43,60 | 60,33 | 0,36 | 0,32 | 228,75 |
| 13 | 9 | 298 | 4,9118 | 4,9166 | 0 | 43,60 | 60,33 | 0,27 | 0,38 | 235,50 |
| 14 | 9 | 313 | 4,9130 | 4,9078 | 1 | 53,90 | 60,34 | 0,69 | 0,72 | 1050,75 |
| 15 | 14 | 328 | 4,9054 | 4,9034 | 1 | 53,23 | 59,94 | 0,65 | 0,66 | 672,00 |
| 16 | 15 | 344 | 4,8996 | 4,8782 | 1 | 52,70 | 57,97 | 0,42 | 0,53 | 731,25 |
| … | … | … | … | … | … | … | … | … | … | … |

**Tab. 2: Structure of tracking result text file.**

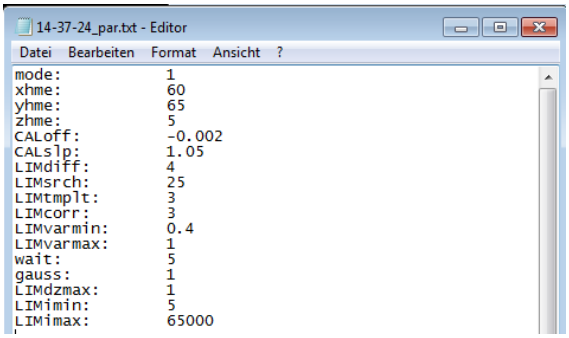



**Fig. 7: Parameter file output.**

A second file with the extension _par.txt lists the tracking parameters set in *param_3DtrackCOR.txt* as well as the template information. The files are named according to the end time of the experiment and placed in a folder named according to the date of the experiment.

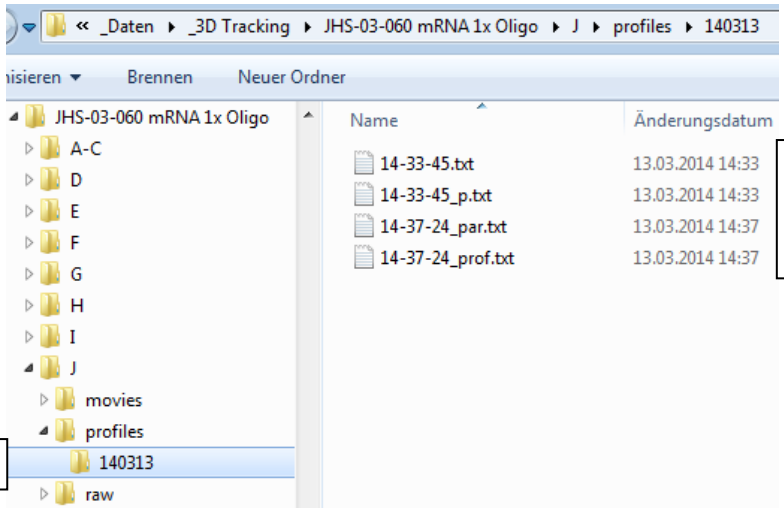
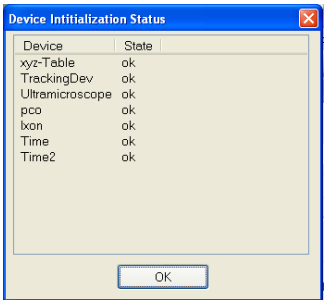



**Fig. 8: Output files.**

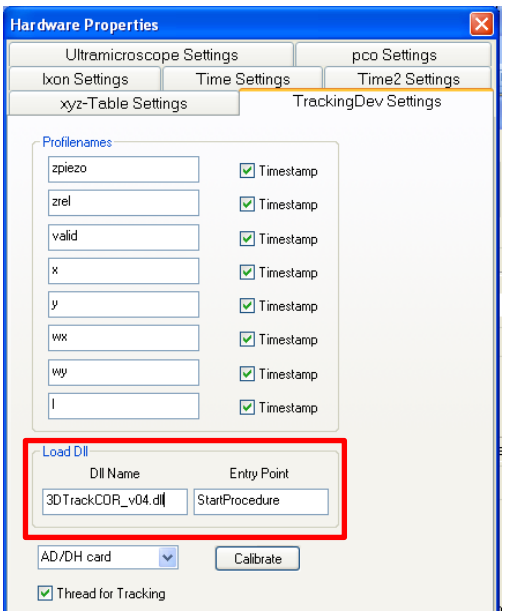
## 7. Performing a tracking experiment in ImSpector

The following figures demonstrate the workflow for performing a feedback tracking experiment using ImSpector.

a) Start program loading the appropriate modules.

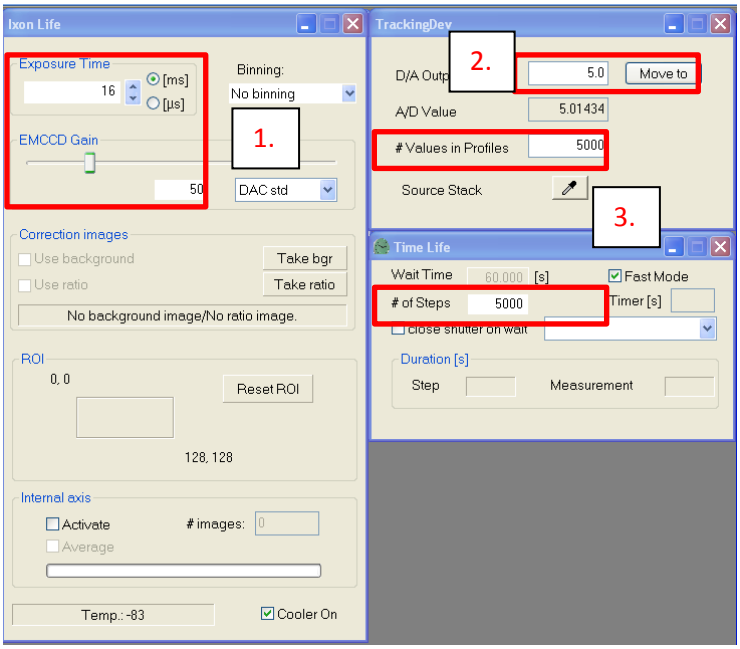



b) Specify DLL containing tracking code in the hardware configuration menu. Save settings and restart ImSpector to load the DLL with the program. Enter names in all eight Profilenames fields and tick all eight instances of Timestamp to initialize the arrays *profiles* and *timestamp* with the appropriate number of columns (8).

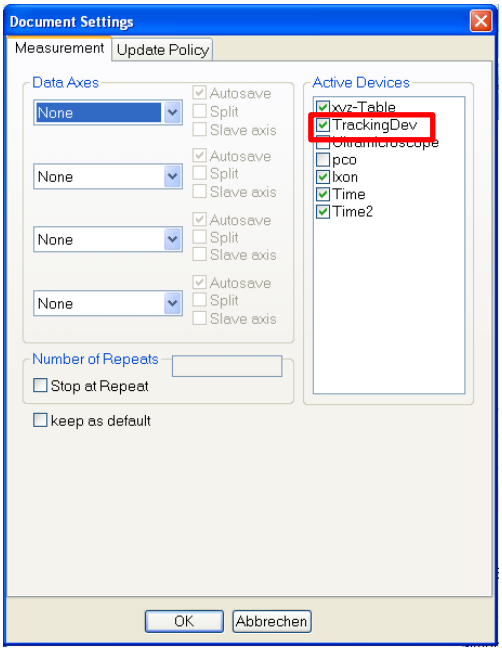


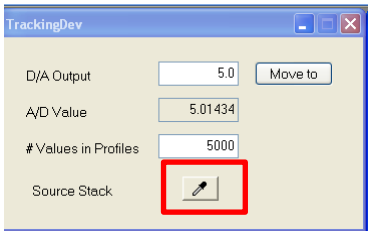

c) Set camera parameters and experiment details (e.g. exposure time, EM gain, series length, …).
Send piezo to center position (5V).
Find region of interest within specimen and specify coordinates for initial particle search in *param_3DtrackCOR.txt*.

"#Values in Profiles" specifies the number of rows in the pre-allocated arrays. Should match the "# of Steps" in the time series to avoid errors!

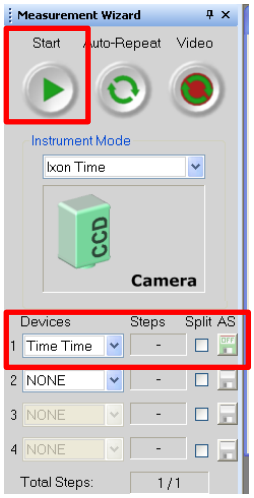


d) Activate tracking in the measurement settings menu. Link TrackingDev to appropriate image stream using the pipette tool.




e) Select time series as active device, adjust auto save settings and click the Start button to run the tracking experiment.



## 8. Contact and additional information

The DLL was written and compiled in Microsoft Visual C++ 2010 Express. It was tested with ImSpector v5.5 using the TrackDev module. Tracking was successfully performed using Andor Ixon DU860 and DV897 EMCCD as well as the pco.edge sCMOS camera. Regular tracking data without feedback loop were also acquired with the Hamamatsu Orca Flash 4.0 and 4.2 camera. However, drivers for these cameras are not incorporated in ImSpector yet.

The TrackDev module is available from LaVision BioTec. The tracking DLL described here was developed by Jan-Hendrik Spille, Institute for Physical and Theoretical Chemistry, University of Bonn, Wegelerstr. 12, 53115 Bonn, Germany. Email: spille@uni-bonn.de

## 9. Version history

141117 – v1.0: Original code as used in the publication.