General information

1.1 License

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.

1.2 Requirements

This software requires an installation of Matlab (MathWorks, Natick, MA) with the following toolboxes:

- Curve Fitting Toolbox
- Image Processing Toolbox
- Optimization Toolbox
- Statistics Toolbox
- Parallel Computing Toolbox (optional)

The software has been tested using the following configuration(s):

- Matlab version: 2011a and above
- Operating systems (64-bit only): Mac OS X 10.8.3, Ubuntu 12.04, Windows 7

1.3 Installation

1. Download the latest version of the software from http://lccb.hms.harvard.edu/software.html
2. Extract the downloaded .zip archive into the directory ‘cmeAnalysisPackage’
3. Start Matlab
4. Add the ‘cmeAnalysisPackage’ directory containing the code to the Matlab search path.
   For more information, see
   http://www.mathworks.com/help/matlab/matlab_env/what-is-the-matlab-search-path.html

This software package contains two main functions, *cmeAnalysis.m* for data analysis and *cmeDataViewer.m* for data visualization. This section briefly describes their use; for further details and a list of additional functions, see section 3.

Documentation for all functions provided with this software can be retrieved by entering *help* followed by the function name in the Matlab command prompt.

2.1 Data organization

The software requires the data organization illustrated in Fig. 1 for automatic parsing:

- Each movie should be stored in a directory with a common identifier, i.e., “Cell”, followed by a number. If the directory name also contains the frame rate, i.e., “Cell1_1s”, this is automatically parsed.
- The frames of each movie must be stored as individual TIFF files (extension .tif or .tiff, case insensitive). If there are multiple channels, each channel must have its own subdirectory within the “Cell” directory. The master channel frames can optionally be stored directly in the “Cell” directory.
- Frames must have a frame number preceding the extension in the file name. There are no constraints on the remainder of the file name.
- Movie directories may be stored in a single parent directory, or further divided by experiment or acquisition session (see Fig. 1). The software can be run at the ‘Condition’ or ‘Experiment’ level; in both cases all movies are pooled for the analysis.

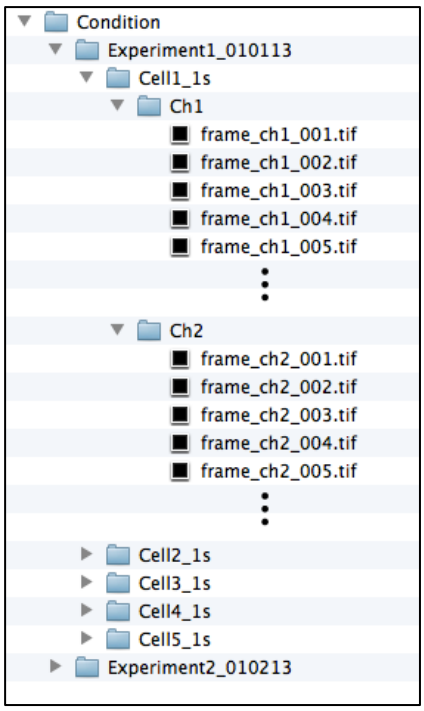

Figure 1 Data structures recognized by the software

2.2 Data analysis

The software takes advantage of Matlab’s parallelization capabilities. To enable this, type

```
>> matlabpool
```

in the command prompt after launching Matlab. Note that this requires the Parallel Computing Toolbox.

To perform the analysis of a set of movies, type

```
>> cmeAnalysis
```

in the Matlab command prompt. This function will ask the user for acquisition parameters and subsequently for the data location. The following acquisition parameters are required for the Gaussian point spread function (PSF) model used for CCP detection: numerical aperture (NA) and magnification of the objective used, and the physical pixel size of the camera (in μm).
After completion of the data analysis, the function will generate plots with the CCP lifetime distribution and intensity cohorts.

The data may be loaded separately and passed to *cmeAnalysis*:

```
>> data = loadConditionData();
>> cmeAnalysis(data);
```

Alternatively, the data can be loaded non-interactively (see also Section 2.3):

```
>> data = loadConditionData('PathToData', {'Ch1', 'Ch2'}, {'EGFP', 'RFP'},...
'Parameters', [1.49 100 6.45e-6]);
```

Using the example of Fig. 1, *PathToData* would point to “Condition”.

The analysis results are returned as data structures:

```
>> [res, data] = cmeAnalysis();
```

where *res* contains the analysis results and *data* contains fields describing and pointing to the movies.

cmeAnalysis must be run separately on groups of movies from different experimental conditions (i.e., control vs. perturbation), since the automatic thresholds for identifying *bona fide* CCPs need to be determined on control data. For such comparisons, first run

```
>> [resCtrl, dataCtrl] = cmeAnalysis;
```

followed by

```
>> [resPert, dataPert] = cmeAnalysis('ControlData', resCtrl);
```

In the first run, select the parent directory of the control data. In the second run, select the parent directory of the perturbation condition.

To view intermediary results of the analysis, including intensity normalization and the distributions used for automated thresholding, include the option

```
>> cmeAnalysis('PlotAll', true);
```

when running the analysis.

2.3 Example data

An example data set is available for download at http://lccb.hms.harvard.edu/doc/cmeTestDataSet.zip
The movies provided are cropped versions of a subset of the EGFP-CLCa O/X movies used in the analysis of Fig. 1-3 in Aguet et al., Dev Cell 2013.

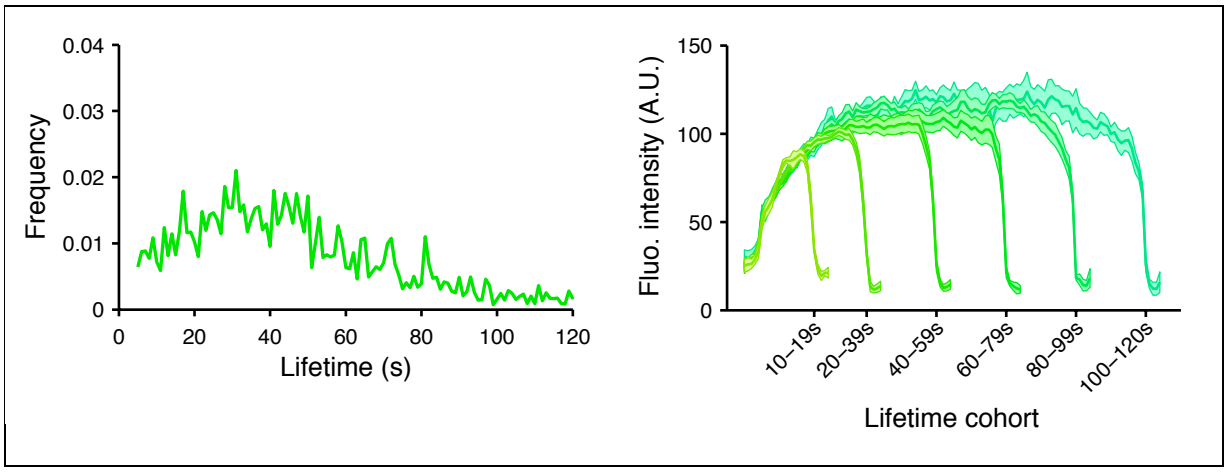

Figure 2 Analysis output for the example data set, showing lifetime distribution and intensity cohort plots

To run the analysis for these movies, either type

```
>> cmeAnalysis
```

and enter the following parameter values: NA 1.49, magnification 100, pixel size 6.45, number of channels: 1, fluorescent marker: EGFP. Alternatively, the data may first be loaded with

```
>> data = loadConditionData('FullPath/testDataSet/', {''}, {'EGFP'},...
'Parameters', [1.49 100 6.45e-6]);
```

and the analysis run with

```
>> cmeAnalysis(data);
```

This will generate lifetime distribution and intensity cohort plots, as illustrated in Fig. 2. To further visualize the results of the analysis for a specific data set, type, i.e.,

```
>> cmeDataViewer(data(1));
```

See also Section 2.5.

2.4 Parameter sensitivity

The software relies on several parameters that may vary as a function of the imaging setup and the acquisition conditions used.
Specifically, the point source detection algorithm used for CCP detection estimates the Gaussian PSF s.d. based on an accurate model of a total internal reflection fluorescence (TIRF) microscope PSF, which in turn is calculated by user-provided parameters (NA and magnification of the objective, camera pixel size, and fluorophores used). In the case of misestimation of any of these parameters or non-ideal TIR conditions, the resulting Gaussian PSF model may be sub-optimal for detection, and a data-derived parameterization of this model may be preferable. To use this data-based estimation, use the option

```
>> cmeAnalysis(..., 'GaussianPSF', 'data');
```

In the tracking step of the analysis, the two most important and sensitive parameters are the maximum number of consecutive missed detection, or “gaps”, in a trajectory, and the search radii for frame-to-frame linking of the detections and for gap closing. These parameters can be adjusted with the options

```
>> cmeAnalysis(..., 'TrackingGapLength', value);
```

and

```
>> cmeAnalysis(..., 'TrackingRadius', [minRadius maxRadius]);
```

Note that these parameters are sensitive to the imaging frame rate and should be adjusted accordingly. The default values are 2 and [3 6], respectively, recommended for data acquired at 0.5 frames/sec (see Extended Experimental Procedures in Aguet et al., Dev Cell 2013). If no changes to these defaults are necessary, then the options can be left out from the function call. If more control over the tracking parameters is needed, the tracking step can be run as described in Section 3.

2.5 Data visualization

Individual data sets may be visualized using

```
>> cmeDataViewer(data(i));
```

where *i* is a numerical index. This launches a graphical user interface displaying the movies with the options to overlay detection and tracking results, and to visualize individual CCS trajectories.
To select between all detected trajectories and the trajectories validated for lifetime and downstream analyses, use the options

```
>> cmeDataViewer(data(i), 'Trajectories', 'all');
```

(default), or

```
>> cmeDataViewer(data(i), 'Trajectories', 'valid');
```

Additional Functions

3.1 Analysis functions

If more control over individual steps in the analysis process is desired, the functions called by *cmeAnalysis* may be run individually. The functions need to be run in order, as each automatically recognizes the output of the previous steps. For further information on each function, see the function documentation in Matlab. All functions require the data structure input returned by *loadConditionData*.

Step 1: Detection

```
>> runDetection(data);
```

Step 2: Tracking

The function *runTracking* is a wrapper for the u-track software, and the function *runTrackProcessing* converts and processes its output.

```
>> settings = loadTrackSettings('Radius', [min max], 'MaxGapLength', value);
>> runTracking(data, settings);
>> runTrackProcessing(data);
```

The *loadTrackSettings* function provides control over all tracking parameters and can be modified if needed.

Step 3: Slave channel classification (only for multi-channel data)

```
>> runSlaveChannelClassification(data);
```

Step 4: Lifetime Analysis

```
>> lftRes = runLifetimeAnalysis()
```

This function identifies the population of *bona fide* CCPs in the data and produces their lifetime distribution curves.

3.2 Graphical output

In addition to the graphical output of *cmeAnalysis* (lifetime distribution and intensity cohort plots) and the data visualization interface *cmeDataViewer*, the following functions are provided:

```
>> plotSNRDistribution(data(i));
```

This displays the PSNR distribution of all detections in the specified data set.

```
>> plotLifetimeComparison({lftRes1, lftRes2, ...});
```

This generates a plot overlaying the lifetime distributions from different conditions. The *lftRes* structures are the outputs of *runLifetimeAnalysis*. When using *cmeAnalysis* on data from two conditions, i.e.,

```
>> [resCtrl, dataCtrl] = cmeAnalysis;
>> [resPert, dataPert] = cmeAnalysis('ControlData', resCtrl);
```

the function call becomes

```
>> plotLifetimeComparison({resCtrl.lftRes, resPert.lftRes, ...});
```

3.3 Analysis results

Analysis results are stored in the *Detection*, *Tracking*, and *Analysis* directories for each movie, within the master channel directory. The function

```
>> loadTracks(data)
```

can be used to import tracks for further analysis.

3.4 Data management

The functions listed in this section provide additional control over data management and loading. See function documentation in Matlab for usage details.

```
>> stk2tiffDirs
```

Splits movies stored as TIFF stacks with extension .tif, .tiff, or .stk into folders containing individual frames.

```
>> selectConditionData
```

Based on *loadConditionData*, enables the selection of individual movies stored in different locations.