

SMALL-LABS User Guide

Benjamin P. Isaacoff, Stephen A. Lee, and Julie S. Biteen

Last updated: September 27, 2018

Table of Contents

1	Introduction	1
1.1	Workflow	1
1.2	The Measurement results	2
2	Using <i>SMALLLABS_main.m</i>	3
2.1	Inputs	3
3	Average Subtraction	4
3.1	Optional parameters	4
3.2	Considerations	4
4	Guessing	5
4.1	Optional parameters	5
4.2	Checking the guessing	5
4.3	Considerations	6
5	Constructing the off-frames list	7
5.1	Considerations	7
6	Subtracting and Fitting	7
6.1	Optional parameters	8
6.2	Gaussian fit parameters	8
6.3	Goodfit (false positive) checks	9
7	Tracking	10
7.1	Optional Inputs	10
8	The ViewFits movie	11
8.1	Optional Inputs	11
8.2	ViewFits color scheme	11
9	Speed and Memory Troubleshooting	12
10	Additional Programs Provided	13
11	The modularity of this code	13
A	List of all optional parameters	14
A.1	Actions	14
A.2	AVGSUB parameters	14
A.3	Guessing parameters	15
A.4	Fitting parameters	15
A.5	Tracking parameters	15
A.6	ViewFits parameters	16

B	The goodframe list	16
C	Coordinate System	16
D	Example movie list file	16

*****If you obtained this code (and thus User Guide) from a source other than the Biteen Lab GitHub please visit <https://github.com/BiteenMatlab/SMALL-LABS> to obtain the latest version of the code.**

1 Introduction

SMALL-LABS is an algorithm which allows single molecules to be localized even in the presence of a spatially arbitrarily complex, temporally low-frequency background. Accurate background subtraction is enabled by separating the foreground from the background based on differences in the temporal variations of the foreground and the background due to fluorophore blinking, bleaching, or moving. Because the background that is subtracted does not include the molecule itself or any neighboring single molecules, the localization and intensity measurement of the molecule is extremely accurate. This document is a detailed guide for users on how to use the provided code. **See the Quick Start Guide for a quick introduction to implementing this code.**

This code runs in Matlab (currently tested in versions 2014a through 2017b), and currently takes input movie data as .mat files, TIFF stacks, and Bio-Formats supported formats, with adaptability to add other image types (TIFF stacks are read by *TIFFStack*, see the note at the end of this section on how to install this package). If your data is in an unsupported format, it is easiest to convert it to a .mat file (preferred) or TIFF stack (otherwise you can add other formats to the function *Movie2mat.m* as described in the function).

The function *SMALLLABS_main.m* is the wrapper for the rest of the code, it performs all of the steps in the SMALL-LABS algorithm in the correct order. Simply run *SMALLLABS_main.m* in the Matlab workspace by specifying the directory containing your movies, the three required parameters, and any optional parameters. The various steps can also all be done independently using *SMALLLABS_main.m* by setting the appropriate action parameters. Furthermore, *SMALLLABS_main.m* can be used to do localization without background subtraction if desired.

1.1 Workflow

The basic (default) workflow (doing full background subtraction) in SMALL-LABS is:

0. Convert the movie file to .mat, if it is in a different format, using *Movie2mat.m*
1. Average (or median) subtraction (make the avgsub movie) using *AVGSUB_movs.m*
⇒ Subtracting this background removes the low-frequency background
2. Molecule detection (guess molecule locations) in the avgsub movie, using *Guessing.m*
3. Identify frames before and after each guess in which there is no other guess nearby, using *Mol_off_frames.m*
⇒ This process gives an *off_frames* list for each guess
4. For each guess, time-average (or median) a small area around the guess using the *off_frames* list, using *Subtract_then_fit.m*
⇒ This average is the True Background

5. Subtract the True Background, and fit the background-subtracted image, using *Subtract_then_fit.m*

Additional optional steps currently included in the SMALL-LABS package:

1. Track the fit results
2. Make a View-Fits movie that shows the fitting results.

1.2 The Measurement results

Several files containing the results from the various steps will be written to the data directory. Most importantly, the fit results will be collected in a .mat file called *moviename_AccBGSUB_fits.mat* which contains the `fits` structure. Each individual guess will generate a row in the `fits` struct, the fields are:

<code>frame</code>	frame number of the fit
<code>molid</code>	the molecule ID as defined in <i>guessing.m</i>
<code>row</code>	row coordinate of the fit
<code>col</code>	column coordinate of the fit
<code>widthr</code>	Gaussian standard deviation in the row dimension
<code>widthc</code>	Gaussian standard deviation in the column dimension
<code>ang</code>	angle of an asymmetric Gaussian fit
<code>offset</code>	intensity offset of the fit
<code>amp</code>	amplitude of the fit
<code>err</code>	error on the fit
<code>chi_squares</code>	if <i>gpufit</i> was used, the final value of the estimator function
<code>sum</code>	sum of pixel intensities in ROI around the fit
<code>rowCI</code>	the 95% confidence interval on <code>row</code> , or if <i>gpufit</i> was used, the estimated localization precision ¹
<code>colCI</code>	the 95% confidence interval on <code>col</code> , or if <i>gpufit</i> was used, the estimated localization precision ¹
<code>widthrCI</code>	if <i>gpufit</i> was not used, the 95% confidence interval on <code>widthr</code>
<code>widthcCI</code>	if <i>gpufit</i> was not used, the 95% confidence interval on <code>widthc</code>
<code>angCI</code>	if <i>gpufit</i> was not used, the 95% confidence interval on <code>ang</code>
<code>offsetCI</code>	if <i>gpufit</i> was not used, the 95% confidence interval on <code>offset</code>
<code>ampCI</code>	if <i>gpufit</i> was not used, the 95% confidence interval on <code>amp</code>
<code>goodfit</code>	Boolean of whether or not the fit is a goodfit
<code>states</code>	if <i>gpufit</i> was used, convergence flag
<code>roinum</code>	the value of the region of interest if a phase mask was used, ² otherwise 1

¹Localization precision based on Thompson R. E.; Webb, W. W.; *Biophys. J.*, **2002**, 82, 2775–2783.

²A phase mask can be a natural number array where a region of the same value is considered a separate region of interest from a region with a different value. If no guesses are to be made in a particular region, the value should be zero.

2 Using *SMALLLABS_main.m*

SMALLLABS_main.m is the wrapper function for the SMALL-LABS algorithm.

Note: that by setting 'bgsb' to false *SMALLLABS_main.m* will do fitting without doing background subtraction. Doing this obviates a lot of parameters, in which case it does not matter what they are set to, and importantly, the required parameters of avgwin and moloffwin can be set to anything in this case.

2.1 Inputs

SMALLLABS_main.m has four required inputs:

file_or_directory_name is the name of the directory where the movies will be selected OR the filename of a single movie OR a list of movies to be fit. If a directory is entered, then *uigetfile* will open and allow movie files (of any extension) to be selected. If the filename of a single movie is entered, then SMALL-LABS will analyze that movie. If a list of movies is entered, then SMALL-LABS will sequentially analyze the movies in that list. To indicate a list of movies, a .txt file with each full filename (e.g., directory\movienamename.ext) on a separate line.

dfrlmsz is the size of a diffraction limited spot in pixels. It is the nominal diameter, which is the integer value of the full width at twenty percent maximum of the Gaussian estimated PSF. dfrlmsz must be an integer! The theoretical diffraction limit of an imaging system, d , is given by $d = \frac{\lambda}{2NA}$, where λ is the emission wavelength maximum and NA is the numerical aperture of the objective. For a given imaging system, we find that a value of $dfrlmsz \approx 1.7d$ in units of pixels works well.

avgwin is the length of the temporal window (in frames) to be used for the average subtraction.

moloffwin is the length of the temporal window (in frames) to be checked to determine in which frames that molecule was off and are thus safe to subtract. Needs to be an even integer.

The inputs are entered with the order as shown below

```
>>SMALLLABS_main(file_or_directory_name, dfrlmsz,  
avgwin, moloffwin, 'optional_parameter_name',  
optional_parameter_value)
```

SMALLLABS_main.m also takes a number of optional parameters (a complete list can be found in Appendix A). Further details about the function of each optional parameter can be found in their corresponding section throughout this guide. All optional inputs are input as name-value pairs, for example to set 'bpthrsh' to 93, include 'bpthrsh', 93 in the function call to *SMALLLABS_main.m*. NOTE: if you reference the default values in *SMALLLABS_main.m* you only need the field name and not the name of the struct, e.g., 'offset', 2500 NOT 'params.offset', 2500.

For example, the function call

```
>>SMALLLABS_main('Test data and simulations', 7, 300,  
100, 'check_guesses', true)
```

opens the *Test data and simulations* directory to select movies there. dfrlmsz=7, avgwin=300, moloffwin=100, and the optional parameter 'check_guesses' is set to true.

If no checking commands (for example 'check_guesses') were sent, *SMALLLABS_main.m* will automatically proceed through the entire calculation of the SMALL-LABS algorithm. After each step, the resultant data is saved in the data directory. A progress bar shows the overall progress

for the current batch of movies and the displays what step is currently being run on which movie; this information is also logged (with timestamps) in the Matlab command window.

In general, SMALL-LABS is a fairly fast and memory-efficient program. This is especially true if a CUDA-enabled GPU is available and/or if Matlab’s parallel computing toolbox is available. If you run into speed or memory problems, see Sec. 9 for strategies to overcome this.

3 Average Subtraction

This step is skipped if `'bgsub'` is set to `false`.

This step is skipped if `'makeAvgsub'` is set to `false`.

The first step in SMALL-LABS is approximate background subtraction by subtracting a moving temporal mean (or median). This is accomplished in *AVGSUB_movs.m*. The parameter `avgwin` is the length of the temporal window over which the mean (or median) is calculated. Because this step can be somewhat slow, first there is a check to determine if the `avgsub` movie has already been created. If so, then there is a further check to determine whether or not any of the parameters have changed. If they are identical, this step is skipped. To bypass this process, simply delete the `avgsub` movie from the directory.

The `avgsub` movie is saved in the data directory as the original movie file name, but with “_avgsub” appended.

3.1 Optional parameters

`'do_avg'` is a Boolean parameter.³ Set to `true` to subtract the average, or set to `false` to subtract the median. The default is `true` (subtract the mean).

`'offset'` is a constant offset to be added to all pixels in the `avgsub` movie after the subtraction. This is done to avoid negative pixels. Because the `avgsub` movie is only used for guessing, the absolute value of these pixels is not very important and `'offset'` should be large enough to avoid negative pixels. The default value is 1000.

3.2 Considerations

The choice of `avgwin` should balance the characteristic on time for the molecules and the characteristic timescale for background changes. `avgwin` should be much longer (at least approximately one order of magnitude longer is ideal) than the characteristic on time of the molecules to minimize the amount of the molecule intensity being subtracted and affecting the guessing. Conversely, `avgwin` should be shorter than the characteristic time of the low frequency background changes, so that over the course a window, the background is essentially constant.

You can check if you chose an appropriate `avgwin` value qualitatively by watching the `avgsub` movie (which is shown during the guess checking procedure): if you see that during part of a molecule’s track it appears as a dark spot instead of a bright spot, then you know that you need to increase `avgwin`. Conversely, if you see that the background features appear and disappear instead of never being present, then you know that you need to reduce `avgwin`. If large portions of the movie look saturated, or if there were warnings printed to the command window about negative pixels, then you need to increase `offset`.

³Boolean variables are either true or false. In Matlab they can be set using `true` or `false`, or 1 or 0.

Setting `'do_avg'` to `false` causes the median to be subtracted instead of the average. This is actually a more accurate way to calculate the background because the median is less sensitive to outliers (in this case molecules appearing). However, because calculating a median first requires sorting, it is slightly slower than calculating a mean, so if you have the extra time or computing power, using the median would be advantageous, but it will cause this step to take longer.

4 Guessing

This step is skipped if `'guessing'` is set to `false`.

The next step in SMALL-LABS is molecule detection, called here guessing.⁴ This is the only step in which molecules are detected, and though there are a number of false positive checks later, there are no further false negative checks. So it is critical to ensure that as many molecules are detected as possible.

4.1 Optional parameters

`'bpthrsh'` is the main parameter to increase or decrease the sensitivity of the detection during guessing. `'bpthrsh'` sets the intensity threshold over which molecule-shaped groups of pixels will be searched. `'bpthrsh'` is the value of the percentile of intensities in the movie (or individual frame, see `'pctile_frame'` below) after the movie was spatially bandpassed to suppress pixel noise. Default is 95.


`'egdesz'` is the number of pixels around the edge of the frame to ignore. Default is `dfrlmsz`.

`'pctile_frame'` is a Boolean that indicates whether or not to calculate the intensity percentile for the threshold (using `'bpthrsh'`) frame by frame, or using the entire movie. If set to `true` the percentile is calculated frame by frame, if set to `false` the entire movie is used. Default is `false`, which is appropriate when contrast is similar for all imaging frames.

`'mask_fname'` is the filename of a mask to set the boundaries of where guessing will occur in frame. To not use a mask simply leave `'mask_fname'` empty (in Matlab, empty is `[]`). If `'mask_fname'` is set `true` (or 1), then the program will look for a file in the same directory as the movie with “_PhaseMask” appended to the name of the movie. See Sec. 4.3 for more details. Default is `[]`, no mask.

`'make_guessmovie'` is a Boolean that indicates whether or not to save a .avi movie showing the guess results to the data directory. If set to `true` the movie will be made and saved, if set to `false`, this step will be skipped. Default is `false`.

4.2 Checking the guessing

It is highly recommended that the guessing parameters are checked for every new movie analyzed (or at least for one or several representative movies in a batch of movies). To do this, set `'checkGuesses'` to `true`. This causes *Guessing.m* to pause in the Matlab debugger once guessing has finished for the first frame. A figure appears showing those results, as shown in Fig. 1. Look through the movie frames to check the guessing. This is accomplished either by pressing the *Continue* button  in the editor tab, or entering `dbcont` in the command window. When ready to

⁴Called guessing for two reasons. Firstly, because the molecule location is imprecisely measured as compared with later super-resolution fitting. Secondly, because the molecules have not gone through a series of false positive checks yet to be sure that it is truly a molecule.

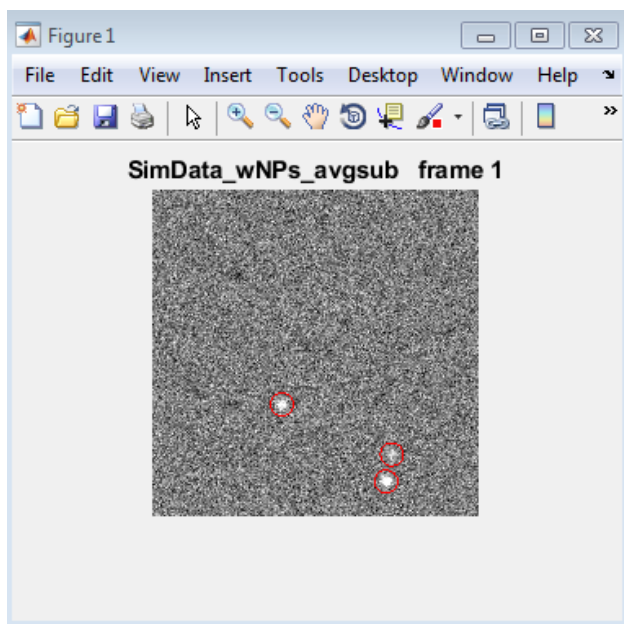


Figure 1: Guessing results as shown in the check guess procedure.

stop checking the guesses, stop the debugger, either by pressing the *Quit Debugging* button  or entering `dbquit` in the command window.

If satisfied with guessing rate, then simply rerun SMALL-LABS with `'checkGuesses'` set to `false`. If not satisfied, change the relevant parameter and rerun SMALL-LABS with `'checkGuesses'` still set to `true`.

In SMALL-LABS it is best to over-guess (include more false positives) slightly. The reason for this is that there are a series of false positive checks that occur later, but no further chance to reduce the false negative rate, so it is best to get it as low as possible here. Furthermore, the accuracy of the true background removal in SMALL-LABS depends on not including images of the molecules in the foreground. So if the false negative rate is too high, images of molecules will be included in the background and reduce the accuracy of the subtraction.

The easiest way to change the guessing rate is by increasing or decreasing `'bpthrsh'`. To increase the detection rate, lower `'bpthrsh'`, and to decrease the detection rate increase `'bpthrsh'`.

If you see that `guessing` is missing a lot of molecules and reducing the threshold does not help discriminate molecules from pixels, then consider changing `dfrlmsz`.

If you are having a hard time determining if guessing is going well or not, try outputting a saved a movie by setting `'make_guessmovie'` to `true`. This enables you to inspect the movie more closely than with the simple frame viewer in SMALL-LABS. We recommend using ImageJ to view the `.avi` movie, which allows easy navigation through the frames and easy scaling of the grayscale.

4.3 Considerations

For SMALL-LABS, it is best to have a *slightly* high false positive rate in guessing for the reasons discussed above. If the false positive rate gets too high though, then it will obviously take more time to analyze all of those guesses. Therefore it is important to balance accuracy here with speed here

by setting a reasonable false positive rate. In practice a false positive rate $\approx 10 - 25\%$ is probably a good target.

If the movie being analyzed has large variations (in molecule density, brightness, background, etc.) over the course of the movie, then it is probably best to guess frame by frame, by setting `'pctile_frame'` to `true`. If the variations are small though, it is likely more robust to use the entire movie to calculate the percentile.

The mask file is a `.mat` file which has a matrix ($m \times n$) called `PhaseMask` that is the same size as a frame ($m \times n$) in the current movie. SMALL-LABS will ignore pixels set to zero in `PhaseMask` for guessing. Pixels in `PhaseMask` set to any other value are grouped together as having a `roinum` of that value. Later, in tracking, SMALL-LABS will only organize molecules in the same track if they have the same `roinum`.

5 Constructing the off-frames list

This step is skipped if `'bgsub'` is set to `false`.

This step is skipped if `'makeOffFrames'` is set to `false`.

The next step in SMALL-LABS is to determine the off-frames list for each detected molecule. This is accomplished in `Mol_off_frames.m`. For each detection, a local off-frames list is constructed; this list enumerates all frames in which no molecule was detected in the local region around the molecule. A local region is defined as a square around the detection with edge size $2 \times \text{dfrlmsz} + 1$ (in pixels). The off-frames list is only constructed for a temporal window around the detection. The length of this temporal window is set by `moloffwin`. There are no optional parameters for this step.

5.1 Considerations

A warning will print to the command window if a detection has fewer than 5% off frames in its `moloffwin`. If you get this warning at all, or especially if you see it for many molecules, consider increasing `moloffwin`.

The same discussion about choosing the length of `avgwin` applies to choosing `moloffwin`. Briefly, it should be as long as possible, without including low frequency changes in the background. Basically, the characteristic on/off time of the molecules should be small compared with `moloffwin`, but the characteristic time for the background to change should be large compared with `moloffwin`.

6 Subtracting and Fitting

This step is skipped if `'fitting'` is set to `false`.

The subtraction of the true background and the PSF fitting are accomplished in the same step in SMALL-LABS: in the `Subtract_then_fit.m` function. If `'bgsub'` is set to `false`, then subtraction is skipped and just fitting is done. The subtraction calculates the mean (or median) of the off-frames (as determined in `Mol_off_frames.m`) in a local region around the guess. The local region is the same $2 \times \text{dfrlmsz} + 1$ (in pixels) square around the guess as was used in `Mol_off_frames.m`.

After fitting, a number of false positive checks are performed. The results of these false positive checks determine whether or not the fit is a *goodfit* and should be used for further analyses.

For background subtraction (`'bgsub'` set to `true`), fit results are saved in a `.mat` file in the data directory called `moviename_AccBGSUB_fits.mat`, where `moviename` is the name of the movie being analyzed. If background subtraction is not being used, then the file is called `moviename_fits.mat`.

6.1 Optional parameters

`'MLE_fit'` is a Boolean that indicates whether or not to use MLE (maximum likelihood estimation) fitting or to use least-squares fitting. If set to `true` MLE fitting will be used, if set to `false` least-squares fitting will be used. Default is `false`.

`'stdtol'` is one of the goodfit (false positive filtering) parameters. `stdtol` is the tolerance on the width (standard deviation of the fitted 2D Gaussian). See Sec. 6.3 for a more detailed discussion on the goodfit parameters. Default is 1.5.

`'maxerr'` is the second changeable goodfit parameter. It is the maximum error on the fit, defined using normalized residuals. See Sec. 6.3 for more details. Default is 0.10.

`'do_avgsub'` is a Boolean that indicates whether or not to subtract the mean or the median of the off-frames. If set to `true` the mean will be subtracted, if set to `false` the median will be subtracted. Default is `true`.

`'which_gaussian'` is a flag that indicates which Gaussian function to fit to. Set to 1 to use a symmetric 2D Gaussian, set to 2 to use a fixed-angle asymmetric 2D Gaussian, set to 3 to use a freely rotating asymmetric 2D Gaussian. See Sec. 6.2 for more details. Default is 1, the symmetric 2D Gaussian.

`'fit_ang'` is the angle (in radians) to which the fixed-angle asymmetric 2D Gaussian will be fit. Default is 0.

`'usegpu'` is a Boolean that indicates whether or not to use a CUDA-enabled GPU for the fitting (if one is available). If set to `true` the GPU will be used, if set to `false` the GPU will not be used. Default is `true`.

6.2 Gaussian fit parameters

There are currently three functional form options for the PSF fit. NOTE: because all of the math is here is done on images (arrays) in Matlab, the coordinate system is rows and columns, r & c , as opposed to something potentially confusing like x & y . See the document included in this package titled *Image Coordinates in Matlab and ImageJ.pdf* for more details.

The first functional form is the symmetric 2D Gaussian

$$(1) \quad A \exp \left(-\frac{(r - r_0)^2 + (c - c_0)^2}{2\sigma^2} \right) + B$$

the fit results are:

$$\begin{aligned} \text{row} &= r_0 \\ \text{col} &= c_0 \\ \text{widthr} &= \sigma \\ \text{widthc} &= \sigma \\ \text{ang} &= 0 \text{ (meaningless in this context)} \\ \text{offset} &= B \\ \text{amp} &= A \end{aligned}$$

The second functional form is the fixed-angle symmetric 2D Gaussian, where the angle $\theta = \text{ang}$.

$$(2) \quad A \exp \left(-\frac{((r - r_0) \cos \theta - (c - c_0) \sin \theta)^2}{2\sigma_r^2} + \frac{((r - r_0) \sin \theta + (c - c_0) \cos \theta)^2}{2\sigma_c^2} \right) + B$$

the fit results are:

```

row    = r0
col    = c0
widthr = σr
widthc = σc
ang    = θ = ang (not a fit parameter here)
offset = B
amp    = A

```

The third functional form is the same as (2), except with the angle θ as an additional fit parameter.

Finally, the `sum` is not a fit parameter, but rather it is the sum of all pixel intensities in the background-subtracted image of the molecule (the local area in the square with side length $2 \times \text{dfrlmsz} + 1$).

6.3 Goodfit (false positive) checks

SMALL-LABS does a number of a goodfit checks to keep the false positive rate low. The three non-customizable checks are:

1. Checking that the fit coordinates (`row` & `col`), the amplitude `amp`, or the sum `sum` are all positive. If any are negative, then it is a bad fit.
2. Checking that the amplitude is less than the sum (`amp < sum`). If `amp > sum` then it is a bad fit.
3. Checking that the errors on the localization (`rowCI` & `colCI`) are smaller than the `dfrlmsz`.

The two customizable goodfit checks are:

1. Compare the width of the Gaussian fit, defined by the standard deviation, σ , to an expected width. For a given nominal diffraction limit size `dfrlmsz` (equivalent to the full width at 20% max), the expected width, σ_E , is $\sigma_E = \text{dfrlmsz} / (2\sqrt{2\log 5})$. A goodfit is a width that satisfies the condition $(\sigma_E / \text{stdtol}) \leq \sigma \leq (\sigma_E \times \text{stdtol})$. For an asymmetric Gaussian, $\sigma = (\sigma_r + \sigma_c) / 2$.
2. Check the error on the fit. The error is defined as a psuedo R^2 value. For a fitted function, F , of a dataset, D , we calculate this psuedo R^2 as:

$$R^2 = 1 - \frac{\sum (D - F)^2}{\sum (D - \langle D \rangle)^2}$$

If $R^2 \geq \text{maxerr}$, then it is a goodfit.

The best way to determine if the goodfit checks are accurately discriminating between false positives and true positives is to watch the ViewFits movie, as described in Sec. 8. If the parameters seem too stringent or too lenient, change them and remake the ViewFits movie to check again. The easiest way to do this is to use the provided function *Change_GoodFits.m* which allows a fit's goodfit status to be reevaluated with new goodfit parameters *without* having to re-fit the movie, and thus can proceed much faster. Similarly, one does not need to make the entire ViewFits movie; instead setting `'write_mov'` to `false` puts the ViewFits function into debug mode, which allows you to inspect a small number of frames carefully, and not wait for the entire movie to be made.

7 Tracking

This step is skipped if `'tracking'` is set to `false`.

After fitting, SMALL-LABS calls a single-particle tracking algorithm using the Hungarian method. Note that if a mask with distinct uniquely numbered ROI's (`roinum`) is used, only molecules in the same `roinum` will be organized into tracks together. The tracks are saved to the fits `.mat` file in the `tracks` variable.

To aid in intensity measurements, SMALL-LABS also creates a track filter logical vector for all guesses called `trk_flg`, which indicates if a molecule was successfully organized into a track AND the particular detection was not the first or last frame of the track. This is to eliminate molecules which have reduced apparent intensities because they turned on or off partway through a frame. A Boolean true indicates that the detection passed the track filter, and a false indicates that it did not.

Note: only goodfits are tracked. If you change the goodfits (for instance with *Change_GoodFits.m*), then you need to re-run tracking.

SMALL-LABS saves the tracking results to the `tracks` array in the fits `.mat` file. Each row in the `tracks` array corresponds to a successfully tracked molecule and the columns are:

1. frame number
2. row position (px)
3. column position (px)
4. track number
5. `roinum`
6. `molid` for correlating with the `fits` struct

7.1 Optional Inputs

Tracking in SMALL-LABS uses the function *Track_3D2.m* (written by David Rowland). Further details about the meaning of the track parameters can be found in that function.

`'savetracks'` is a Boolean that indicates whether to save the output of *Track_3D2.m* directly in its own `.mat` file. The `tracks` & `trk_flg` variables will still be saved to the fits `.mat` file. Set to `true` to additionally save the raw output from *Track_3D2.m*, and set to `false` not to. Default is `false`.

The next inputs are the tracking parameters. These are inputted to SMALL-LABS as a single vector. We will only give a cursory explanation of these parameters here; see *Track_3D2.m* for more details. The default values are `trackparams=[0.01,200,0.5,3,3,1,0]`, where:

minimum merit	<code>trackparams(1)</code>
integration time (ms)	<code>trackparams(2)</code>
gamma	<code>trackparams(3)</code>
maximum step size (pixels)	<code>trackparams(4)</code>
minimum track length (frames)	<code>trackparams(5)</code>
speed estimation window halfsize	<code>trackparams(6)</code>
time delay between consecutive frames (ms)	<code>trackparams(7)</code>

8 The ViewFits movie

This step is skipped if `'makeViewFits'` is set to `false`.

Lastly, SMALL-LABS makes a ViewFits movie to show the results of detection, fitting, and tracking. The ViewFits movie is saved as `moviename_ViewFits.avi` in the same directory as the data.

8.1 Optional Inputs

`'orig_movie'` is a Boolean that indicates whether or not to use the original movie, or to use the avgsub movie. Set to `true` to use the original movie, and set to `false` to use the avgsub movie. Default is `true`.

`'circ_D'` is the diameter (in pixels) of the circles shown in ViewFits movie to indicate molecule positions. Default is `dfrlmsz`

`'linewidth'` is the linewidth of the circles. Default is 1.

`'write_mov'` is a Boolean that indicates whether or not to write the ViewFits movie to the data directory, or to go into debug mode and allow the user to click through frames and inspect them there. Set to `true` to make the entire movie and save it to the data directory, or set it to `false` to enter debug mode without saving. Default is `true`.

`'autoscale_on'` is a Boolean that indicates whether or not to set the grayscale frame-by-frame or use a single grayscale for the entire movie. If using the entire movie 100 equally spaced frames are taken from the movie to set the grayscale. Set to `true` set the grayscale frame-by-frame, and set to `false` to use one for the entire movie. Default is `false`.

`'trackingVF'` is a Boolean that indicates whether or not to use the “standard” ViewFits format showing guesses, fits, and the track filter, OR to show only color coded tracks. Set to `true` to make the tracking ViewFits movie, and set to `false` to make the “standard” ViewFits movie. Default is `false`.

8.2 ViewFits color scheme

For the “standard” ViewFits, a green circle indicates a guess that passed the goodfit checks, a red circle indicates a guess that did not pass the goodfit checks, and a pink circle indicates a molecule that passed the goodfits checks and track filter. An example frame of a “standard” ViewFits movie is shown in Fig. 2a

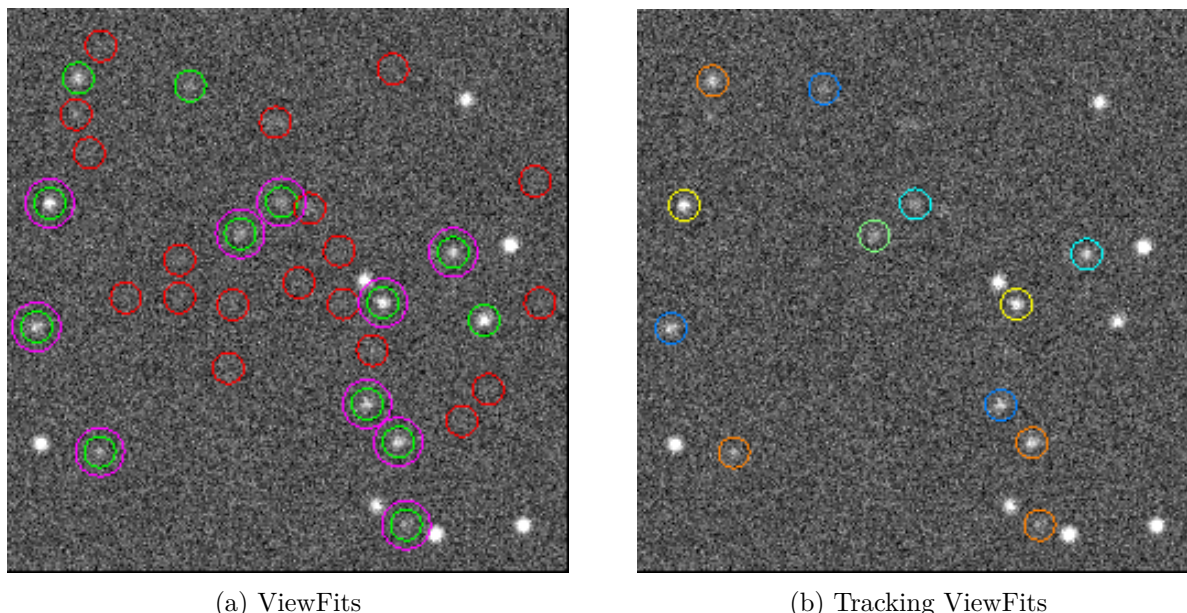


Figure 2: ViewFits frame. Note: this movie had several molecule-shaped objects in the background; those are not missed molecules but actually the background.

For the tracking ViewFits movie, the circles only indicate molecules which were successfully organized into tracks. Colors indicate unique tracks. For simplicity, there are only five unique colors that are cycled through. An example frame (same movie and frame number as in Fig. 2a) of a tracking ViewFits movie is shown in Fig. 2b.

Note: Two molecules in the same frame will never be organized into the same track, so though in Fig. 2a there are multiple molecules in one frame with the same color they are in fact in different tracks and the five-color cycle has been repeated.

9 Speed and Memory Troubleshooting

SMALL-LABS is generally a fairly fast and memory-efficient program, however a number of steps can be taken to make it faster or more memory-efficient.

Use smaller movies. Obviously, analyzing smaller movies will require less memory and time. You can use the provided program *ROI_picker.m* to pick a smaller region of interest (ROI) within your original movies. If you consistently run out of memory in SMALL-LABS, you have to either increase the memory of your computer or analyze the movie as several smaller movies.

Analyze fewer molecules. Every molecule guessed has to be analyzed: this process includes creating an off-frames list, subtracting the True Background, and fitting. Increasing the number of guessed molecules adds a lot more computations. If you reduce the number of guesses, this can greatly reduce the time to run the program. You can accomplish this by increasing `bpthrsh`, but be careful not to increase it so much that you miss molecules. You can also increase `egdesz` to skip fitting molecules near the edge of the frame (which may not be meaningful). Finally, using least squares fitting, setting '`MLE_fit`' to `false`, is much faster than MLE fitting.

Do not make a ViewFits movie. The optional step to make a ViewFits movie is fairly slow. If analyzing a batch of movies, we recommended you do all of the analysis first and then separately make ViewFits movies only for a few representative movies.

Use parallel computing. If you do not have the Parallel Computing Toolbox for Matlab, installing it allows SMALL-LABS to parallelize some steps. Additionally, running SMALL-LABS on a computer with a CUDA-enabled GPU will allow the fitting step to proceed much faster.

10 Additional Programs Provided

This package provides a number of additional programs which are not needed for the main operation of SMALL-LABS. The following are short descriptions of these programs; for additional details see the comments at the beginning of each program.

Change_GoodFits.m is a function which can be run after fitting to change the goodfit parameters (`maxerr` and `stdtol`) as well as change `dfirlmsz`, and consequently, this function changes which fits are considered goodfits. This is much faster because it allows you to experiment with these parameters without having to fit the movie again. Note: this does not change the fitting results such as `row`, `col`, `widthr/widthl`, `amp`, *etc.*

ROI_picker.m is a function which allows you to pick an ROI in a series of movies that are in the TIFF stack format and write a new `.mat` movie of just that ROI. This is helpful because it allows you to reduce the size of the movies being analyzed, which will decrease how long it takes to analyze them.

stack_tiffs.m is a function which converts a folder of individual TIFF images into a single TIFF stack. By using *TIFFStack* and *saveastiff*, it is fairly fast.

WriteListOfMovies.m gives an example of how to write such a list of movies using Matlab to be analyzed as a batch in SMALL-LABS. The beginning section of that script takes advantage of a specific naming convention for the movies (each is named `mov_XXXX.mat`), which may not be appropriate for your data, but it can be a good starting point for writing your own script.

In the Test data and simulations directory, there two programs for generating and working with simulated data:

SimulateDataPoisson.m is a simple script to simulate single-molecule data with various backgrounds. It generates data with Poissonian noise.

Check_Fitting_Simulated.m is a script which can be used to check how accurately simulated data was fit.

11 The modularity of this code

SMALL-LABS is written as modular code. It was designed so that different parts of the code can be easily modified or even completely replaced. The code has been copiously commented and we have tried in particular to explain the inputs and outputs of the various steps and functions. This should help make any modifications or replacements simpler to implement.

In particular, we anticipate users may wish to utilize a specific guessing (detection) method or specific fitting method. For example, if a non-Gaussian PSF is desired, then the provided fitting functions will not be appropriate and the guessing algorithm may not work. By simply modifying those two steps, SMALL-LABS should still function and accurately subtract the local background.

To modify the guessing step, the simplest approach would be to replace the entire guessing function. The function should output the guesses in the same format as the provided function: as a .mat file using the same naming convention, and with an array called guesses that has columns 1. frame number, 2. row position (pixels), 3. column position (pixels).

To modify the fitting step, the simplest approach would be to add another if statement in the fitting section of *Subtract_then_fit.m*. In this new section, fit the same array as in the other sections, called dataset. Then enter the fit results into the fits structure using the column definitions described in the introduction of this guide. Note that the goodfit checks will likely need to be modified to be appropriate for the new fitting method.

A List of all optional parameters

All optional inputs are input as name-value pairs, for example to set 'bpthrsh' to 93 include 'bpthrsh', 93 in the function call to *SMALLLABS_main.m*. This is a complete list of all the optional parameters and their default values; a duplicate list is in the comments in the beginning of *SMALLLABS_main.m*.

SMALL-LABS prints to the command window for every step it takes. If you want to disable this feature, edit *SMALLLABS_main.m* and change the line `verbose=true` (immediately preceding the `%%Parameter Defaults` section) to `verbose=false`.

A.1 Actions

Do background subtraction

'bgsub' = true

Make the average subtracted movie

'makeAvgsub' = true

Do guessing

'guessing' = true

Check the guesses

'checkGuesses' = false

Make the off frames list

'makeOffFrames' = true

Do fitting

'fitting' = true

Do tracking

'tracking' = true

Make the ViewFits movie

'makeViewFits' = true

A.2 AVGSUB parameters

Subtract the temporal average? otherwise use median

'do_avg' = true

AVGSUB offset

```
'offset' = 1000
```

A.3 Guessing parameters

the threshold percentile of the bandpassed movie

```
'bpthrsh' = 95
```

how many pixels to ignore around the edge of the frame

```
'egdesz' = dfrlmsz
```

compare brightnesses in each frame? otherwise use entire movie

```
'pctile_frame' = false
```

use a mask for guessing? Enter its filename or set to true to look for `“_PhaseMask”` appended to movie name

```
'mask_fname' = []
```

make a movie (.avi) of the guesses to check parameters

```
'make_guessmovie' = false
```

A.4 Fitting parameters

do MLE fitting? If not least-squares fitting will be used

```
'MLE_fit' = false
```

Goodfit parameters. See *Subtract_then_fit.m* for the details

```
'stdtol' = 1.5
```

```
'maxerr' = 0.10
```

subtract the mean of off frames? If not use median

```
'do_avgsub' = true
```

Which Gaussian function to fit to when using fitting? 1. symmetric, 2. fixed angle asymmetric, 3. free angle asymmetric

```
'which_gaussian' = 1
```

the angle to fit to for a fixed angle Gaussian

```
'fit_ang' = 0
```

use a GPU if one is available?

```
'usegpu' = true
```

A.5 Tracking parameters

save the separate tracks .mat file?

```
'savetracks' = false
```

`'trackparams'` needs to be entered as a whole vector, the default is [0.01,200,0.5,3,3,1,0]

minimum merit	<code>trackparams(1)=0.01</code>
integration time (ms)	<code>trackparams(2)=200</code>
gamma	<code>trackparams(3)=0.5</code>
maximum step size	<code>trackparams(4)=3</code>
minimum track length	<code>trackparams(5)=3</code>
speed estimation window halfsize	<code>trackparams(6)=1</code>
time delay between consecutive frames (ms)	<code>trackparams(7)=0</code>

A.6 ViewFits parameters

use the original movie? if not, use the avgsub movie

```
'orig_movie' = true
```

diameter of the circles showing the fits

```
'circ_D' = dfrlmsz
```

linewidth of the circles

```
'linewidth' = 1
```

write a .avi movie showing the fits. If not, goes to debug mode

```
'write_mov' = true
```

autoscale frame by frame?

```
'autoscale_on' = false
```

use the tracking viewfits instead

```
'trackingVF' = false
```

B The goodframe list

SMALL-LABS has a built-in functionality to ignore specific frames. This is accomplished using the `goodframe` vector, which is a logical vector whose length is equal to the number of frames in the movie (giving a one-to-one correspondence between each frame in the movie and an element in `goodframe`). Unless given a `goodframe` list, SMALL-LABS assumes all frames are to be used.

To input a `goodframe` list, simply save the `goodframe` vector to the .mat file containing the movie (`mov` array). The `goodframe` vector needs to be a logical vector, meaning either logical type variable or all of its elements must be convertible to logical (0 & 1, etc.).

C Coordinate System

SMALL-LABS runs entirely in Matlab, and as such the coordinate system used is the most meaningful for Matlab arrays. Instead of returning x & y positions in some arbitrary coordinate system, SMALL-LABS uses, and returns, coordinates in row and column position using Matlab conventions for rows and columns and the origin. Throughout most of the program, the row and column numbers (integers) are tracked. The fitting step, which provides sub-pixel information, returns a non-integer row and column position. For example, fitting a guess at $(r, c) = (41, 153)$ may yield a position of $(r, c) = (42.0597, 153.9108)$.

For a more detailed tutorial on the coordinate system in Matlab and how it relates to the “normal” convention used in a Cartesian coordinate system, and also the coordinate system used in ImageJ (a free commonly used image viewer and analysis program), see the pdf included in this repository titled *Image Coordinates in Matlab and ImageJ.pdf*.

D Example movie list file

To send a list of movies to SMALL-LABS, enter the .txt file containing the list as the first input to `file_or_directory_name`. The .txt file should have the full filename (e.g., `directory\movienamename.ext`) for each movie on a separate line. For example, to analyze the following list of movies

list_file.txt

```
E:\Microscope Data\2_29_18\mov_001.mat  
E:\Microscope Data\2_29_18\mov_002.mat  
E:\Microscope Data\2_29_18\mov_003.mat  
E:\Microscope Data\2_29_18\mov_005.mat  
E:\Microscope Data\2_29_18\mov_008.mat  
E:\Microscope Data\2_29_18\mov_013.mat  
E:\Microscope Data\2_29_18\mov_021.mat  
E:\Microscope Data\2_29_18\mov_034.mat  
E:\Microscope Data\2_29_18\mov_055.mat  
E:\Microscope Data\2_29_18\mov_089.mat  
E:\Microscope Data\2_29_18\mov_144.mat
```

simply enter

```
>>SMALLLABS_main('list_file.txt', 7, 300, 100,)
```

The optional script *WriteListOfMovies.m* gives an example of how to write such a list of movies using Matlab. The beginning section of that script takes advantage of a specific naming convention for the movies (each is named mov_XXXX.mat), which will likely not be appropriate for your data, but it can be a good starting point for writing your own script.