

# SMALL-LABS

## User Guide

Benjamin P. Isaacoff  
University of Michigan  
last update: February 28, 2017

# Contents

Introduction.....	1
Using <i>SMALLLABS_main</i> .....	2
Average Subtraction.....	5
Guessing.....	6
Constructing the off_frames list.....	8
Subtracting and Fitting.....	8
Tracking .....	9
ViewFits.....	10
Fitting without background subtraction .....	11
Coordinate System Definition.....	12
Speed and Memory Troubleshooting.....	12
Additional Programs Provided.....	13
The modularity of this code .....	13

**\*\*\*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.**

## 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. Because the background subtracted is the true background (minimizing or eliminating subtraction of molecule images) 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 the using this code.

This code runs in Matlab (currently tested in versions 2014a through 2016b) and currently only takes input data as TIFF stacks (using *TIFFStack* to read the TIFFs, see the note at the end of this section on how to install this package). If your data is in another format, the easiest thing to do would be to convert it to a TIFF stack (otherwise you'd have to edit the code to accept the different format, a difficult, though not impossible task). Conversion to TIFF stack can be accomplished using a variety of programs; in Matlab, this can be accomplished by importing your data to a 3D array (the first two dimensions are spatial and the third dimension is frame number), then using *saveastiff* to save it as a TIFF stack.

The function *SMALLLABS\_main* is the wrapper for the rest of the code, it perform all of the steps in in the SMALL-LABS algorithm in the correct order. Simply run *SMALLLABS\_main* in the Matlab workspace by specifying the directory containing your TIFF stack movies and specifying the three other required parameters and any optional parameters, then run it and click to choose the movies you want to fit. The various steps can also all be done independently, either by using their individual standalone functions, or by using *SMALLLABS\_main* to run them individually by setting the appropriate action parameters.

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

1. Average subtraction (make the avgsub movie) using *AVGSUB\_tiffs.m*
  - ⇒ Subtracting this background removes low frequency background
2. Molecule detection (guess molecule locations) in the avgsub movie using *Guessing.m*
3. With *Mol\_off\_frames.m*, identify frames before or after each guess in which there isn't another guess nearby, for instance because the molecule hasn't turned on yet, it turned off, or it blinked.
  - ⇒ This process gives an off\_frames list for each guess.

4. For each guess, time-average a small area around the guess using the `off_frames` list; this process occurs in *Subtract\_then\_fit.m*
  - ⇒ This time average is the True Background
5. Subtract the True Background, and fit the background-subtracted image using *Subtract\_then\_fit.m*

### OPTIONAL STEPS

6. Track the fits for a variety of purposes, the default being to filter out fits which aren't organized into a track, and remove the first and last frame of each track using *Track\_filter.m*
7. Make a ViewFits movie to help check the fitting using *ViewFits.m*

Several files containing the results from the various steps will be written to the working directory. Most importantly, the fits will be collected in a .mat file called *moviename\_AccBGSUB\_fits.mat* which contains the *fits* structure. Each individual guess will generate a new row in the *fits* fields which are

<i>frame</i>	frame number of the fit
<i>row</i>	row coordinate of the fit
<i>col</i>	column coordinate of the fit
<i>widthr</i>	Gaussian standard deviation in the row dimension
<i>widthc</i>	Gaussian standard deviation in the column dimension
<i>ang</i>	angle of an asymmetric Gaussian fit
<i>offset</i>	offset of the fit
<i>amp</i>	amplitude of the fit
<i>err</i>	error on the fit
<i>sum</i>	sum of pixel intensities in the ROI around the fit
<i>goodfit</i>	Boolean of whether or not the fit is a goodfit

The following sections go through each step in more detail, defining and explaining the various parameters (required and optional) for each step, and the inputs and outputs for that step.

NOTE: The program *TIFFStack* is used to rapidly read tiff stacks. In order to use *TIFFStack*, copy the @TIFFStack folder from this repository (leave it as a folder with that name) into your Matlab directory. The *TIFFStack* README has more details if you have problems.

## Using *SMALLABS\_main*

The four required parameters to *SMALLABS\_main* are:

*directoryname* is the name of the directory in which the movies will be selected. If there is an error finding the directory, the program will open uigetfile in the current working directory

*dfrlmsz* is the size of a diffraction limited spot in pixels. It's the nominal diameter, NOT the FWHM or Gaussian standard deviation. It must be an integer. For an expected diffraction-limited standard deviation, std, using the full width at 20% max,  $dfrlmsz = \text{std} \times (2 \times \sqrt{2 \times \log(5)})$ , is a good approximation.

*avgwin* is the length of the temporal window (in frames) to be used for the average subtraction. Needs to be an odd integer.

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

The order to send these parameters is

```
SMALLLABS_main(directoryname,dfrlmsz,avgwin,moloffwin,varargin)
```

Optional parameters (NOT the required parameters) taken at the varargin position in the function call are sent to *SMALLLABS\_main* using a name-value pair. If you want to specify an optional parameter you include it in the function call as

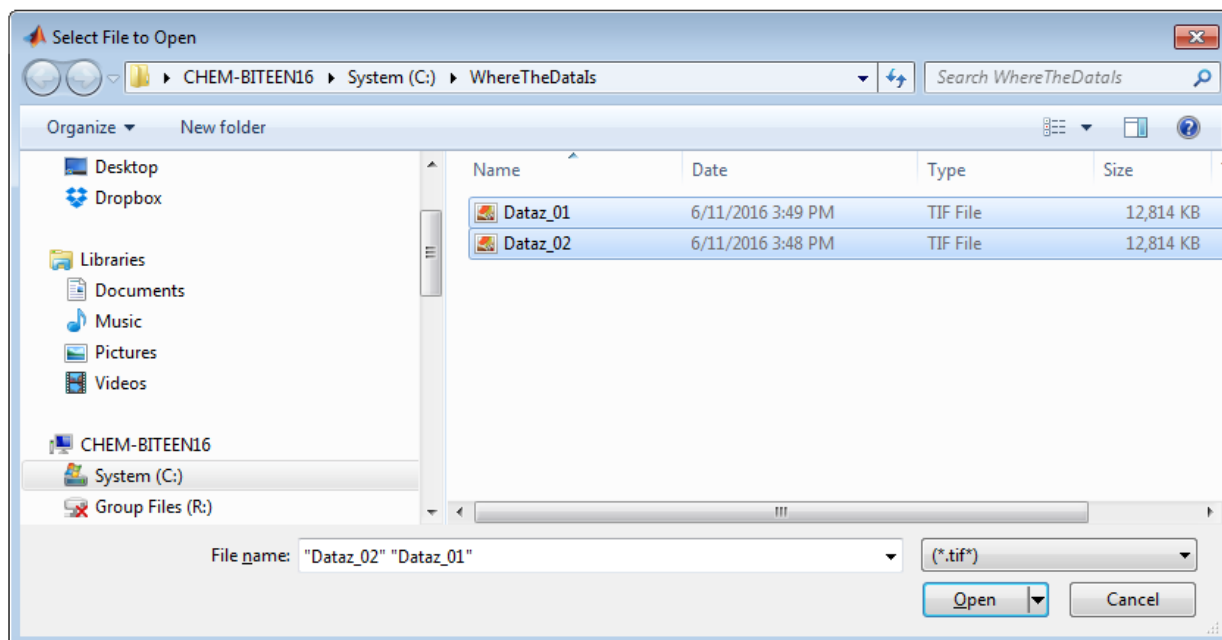
```
'parameter_name',parameter_value
```

The various optional parameters can be found throughout this guide. However, a simpler way to find the exact parameter name and to check what its default value is (and thus if you want to use another value) is by looking in the Parameter Defaults section after the introduction in the *SMALLLABS\_main* code. Note that you only need the field name and not the name of the structure, e.g., 'offset', and NOT 'params.offset'.

For example if you set *dfrlmsz* = 7, *avgwin* = 501, and *moloffwin* = 150, and your data is stored in the directory C:\WhereTheDataIs, and you set the optional parameters *bpthrsh* = 90 and *orig\_movie* = 0, you would enter:

```
SMALLLABS_main('C:\WhereTheDataIs',7,501,150,'bpthrsh',90,'orig_movie',0)
```

This brings up the uigetfile interface for you to select the TIFF stack movies, which on my system, after selecting the two movies in the directory looks like:



*SMALLLABS\_main* will then proceed to calculate and write out the various results without any further input from the user. Many of the steps will display a progress bar to show how far along that particular step is. In general, *SMALLLABS\_main* is a fairly fast and memory-efficient program. The slowest step is fitting, where least squares fitting (the default) is much faster than MLE fitting. If speed is an issue, reducing the number of guesses will reduce the total time required for *SMALLLABS\_main* to run, see the Speed and Memory Troubleshooting section for more details.

*SMALLLABS\_main* can accept many optional parameters, they will be covered in this document (and also in the Parameter Defaults section of the *SMALLLABS\_main* code). You can control which actions *SMALLLABS\_main* does by changing the following Booleans:

*bgsb* determines whether background subtraction occurs. Default is 1.

*makeGuesses* determines whether the avgsub movie will be made (if one doesn't already exist) and if guessing will be done. Default is 1.

*check\_guesses* determines whether the guessing will be checked visually by the user. Default is 0.

*makeOffFrames* determines whether the off frames list will be constructed. Default is 1.

*fitting* determines if the subtract & fitting program will be run. Default is 1.

*tracking* determines if tracking will be run. Default is 1.

*makeViewFits* determines if the *ViewFits* program will be called. Default is 0.

## Average Subtraction

The first step in SMALL-LABS is making the average-subtracted (avgsub) movie. Because this step can be somewhat slow, *SMALLABS\_main* checks if *moviename\_avgsub.tiff* already exists in the working directory. If it does then the program skips making a new avgsub movie. If you want to make a new avgsub movie, simply delete the *moviename\_avgsub.tiff* file from that directory.

If *bgsub* was set to 0, this step will be skipped and guessing will proceed using the original movie.

This step utilizes *AVGSUB\_tiffs.m* which is called like:

```
AVGSUB_tiffs(movie_filename,do_avg,runningavg,avgwin,offset);
```

### *SMALLABS\_main* required parameters

*avgwin* is the length of the temporal window (in frames) to be used for the average subtraction. Needs to be an odd integer.

### *SMALLABS\_main* optional parameters

*movie\_filename* is the filename of the original TIFF stack movie.

*do\_avg* is a Boolean that determines whether the average will be subtracted, otherwise the median will be subtracted. Default is 1.

*runningavg* is a Boolean, set to 1 to do a running average background subtraction or set to 0 to do a static window background subtraction. Default is 1.

*offset* is the intensity offset to compensate for possible negative pixels after subtraction. Default is 1000.

This functions saves the avgsub movie as a TIFF stack in the original file location with *\_avgsub* appended to the original file name. The function also outputs a short text file with the parameters used called *moviename\_avgsub\_info.txt*

Choosing *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 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 smaller 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: 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, then you need to increase *offset*.

Setting *do\_avg* to 0 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 much 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 a lot longer.

## Guessing

Guessing (molecule detection) is accomplished in the function *Guessing.m* which is called like:

```
Guessing(movie_filename,dfrlmsz,bpthrsh,edgesz,pctile_frame,check_guesses,mask_fname);
```

*movie\_filename* is the filename of the movie

### SMALLLABS\_main required parameters

*dfrlmsz* is the nominal size of a diffraction-limited spot in your microscope, the full width at 20% max is a good approximation

### SMALLLABS\_main optional parameters

*bpthrsh* is the percentile of brightnesses of the bandpassed image below which intensity, pixels will be ignored. Default is 95

*edgesz* is the number of pixels on the edge of the image that will be ignored. Default is *edgesz* = *dfrlmsz*

*pctile\_frame* is a Boolean determining whether *bpthrsh* will be applied frame by frame, or to the entire movie. Using the entire movie (setting to 0) is more sensitive to low frequency noise and background changes, it is however a more robust guessing method as the much higher number of molecules that appear throughout the movie make this statistical measure (percentile) more meaningful. Default is 0.

*check\_guesses* is a Boolean to determine if you want to go through and look at the guesses. Default is 0

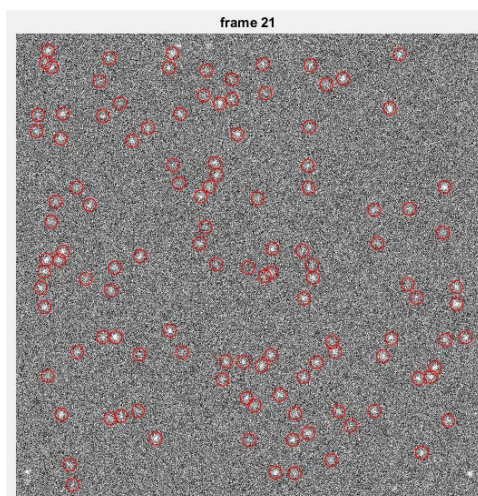
*mask\_fname* is the filename of a mask to use for guessing. If no mask is being used just leave it empty. If *mask\_fname* is set 1, then the program will look for a file in the same directory as the movie with '\_PhaseMask' appened to the name of the movie. The mask is a .mat file which has a logical array (or at least where nonzero entries will be converted to 1s) called PhaseMask that is the same size as a frame in the current movie. Default is no mask.


*make\_guessmovie* is a Boolean to determine if you want to output a .avi movie of the guesses. Default is 0.

*Guessing.m* saves a .mat file called *movienam\_guesses.mat* which contains the *guesses* array. The columns in *guesses* are simply: 1. frame #, 2. row position (pixels), 3. column position (pixels).



You should verify that the guessing is working the first time you run *SMALLLABS\_main*. There are two methods to do this verification. The first method is to simply set *check\_guesses* to 1 when you call the function, and when the program gets to guessing you will see the guesses indicated with red circles overlaid on frames for the movie, for example something like:



You can click through the frames by continuing the debug process either by pressing  (in the editor tab) or by typing *dbcont* into the workspace. If the parameters don't look good, simply stop the debugging and change the parameters when you run it next. The second method is to instead set *make\_guessmovie* to 1, which will output a .avi movie of the guesses. This movie is identical to the individual frames shown when *check\_guesses* is set to 1, however there are a number of potential advantages to being able to watch the movie in an external player (such as ImageJ) instead of frame by frame in Matlab. The two biggest advantages I see are 1. being able to watch the movie at any speed and actively change the contrast settings, and 2. being able to navigate through the movie forwards as well as backwards. Using this method, if the parameters don't look good, simply stop the program and change the parameters when you run it next.

Repeat this process until you find a set of parameters that give good guessing results. The goal is to have the program guess with a low false negative rate (not guessing molecules) and a low false positive rate (guessing when a molecule isn't present). In case you're having a hard time balancing these, for SMALLLABS it is **much better** to have false positives than to have false negatives. However too many unneeded guesses (false positives) will make the fitting step last much longer than it needs to, and can reduce the precision of the fitting result by having a very noisy image subtracted.

The first parameter to try changing is *bpthrsh*. If *Guessing* is missing a lot of molecules (false negatives), then try reducing *bpthrsh*, and conversely, if *Guessing* is producing a lot of false positives, then try increasing *bpthrsh*. Because *bpthrsh* sets the percentile of brightnesses, it is not a brightness value threshold *per se*, and thus it should be fairly insensitive to the magnitude of the brightness in a movie. Rather the optimal value depends mainly on the density of bright objects (technically pixels) in the bandpassed movie.

The other parameters to consider changing to help the guessing are *dfrlmsz* and *pctile\_frame*. If *dfrlmsz* is very off from its true value, then *Guessing* will not be able to find the molecules well. If large low frequency changes occur over the course of a movie, then it would probably be best to turn *pctile\_frame* on.

Note that if the movies being analyzed are very large files, you may have a memory issue in this step if *pctile\_frame* is turned off. If you do have this issue then simply turn *pctile\_frame* on.

## Constructing the off\_frames list

If *bgsb* was set to 0, this step will be skipped as it is unneeded.

The off\_frames list is constructed using the *Mol\_off\_frames.m* function, which is called like:

```
Mol_off_frames(guess_filename,dfrlmsz,moloffwin);
```

*guess\_filename* is the filename of the .mat file output from *Guessing.m* which contains the *guesses* array.

*SMALLLABS\_main\_required parameters*

*dfrlmsz* is the nominal size of a diffraction limited spot for your system, the full width at 20% max is a good approximation

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

*Mol\_off\_frames.m* doesn't have any optional parameters and shouldn't need to be debugged. Briefly, this function identifies frames in which two guesses are within a  $2 \times dfrlmsz$  sized box of each other. A warning will print out to the workspace if a particular guess has less than 5% of *moloffwin* frames off. If you see a lot of this warning you should consider increasing *moloffwin*.

The off frames list will be saved to a .mat file called *guess\_filename\_Mol\_off\_frames.mat*

## Subtracting and Fitting

Subtracting and fitting (steps 4 & 5 in the intro) are accomplished in *Subtract\_then\_fit.m* which is called like:

```
Subtract_then_fit(movie_fname,Mol_off_frames_fname,guess_fname,MLE_fit,egdesz,stdtol,  
maxerr,do_avgsub);
```

*movie\_fname* is the filename of the original TIFF stack movie (NOT the avgsb movie)

*Mol\_off\_frames\_fname* is the name of the .mat file containing the off frames list. If background subtraction is not being done, then this input is set to '*nobgsb*'.

*guess\_fname* is the filename for the guesses .mat file

### SMALLLABS\_main\_optional\_parameters

*MLE\_fit* is a Boolean that determines whether or not MLE fitting is used. Set to 1 to use MLE and to 0 to use least squares using *lsqcurvefit*. Default is 0. Note that MLE is quite slow, and so it is not recommended for a large number of guesses.

*edgesz* is the number of pixels on the edge of the image that will be ignored. Default is *edgesz = dfirlmsz*

*stdtol* is the ratio tolerance on fit Gaussian STD, default value is 1.5. Meaning that if the  $\text{std}/\text{stdtol} \leq \text{fit\_std} \leq \text{std} \times \text{stdtol}$  then it is considered a goodfit.

*maxerr* is the maximum error of the fit. For MLE fit, using variance default 0.1 (can't be above this) for LSQR fit, using the mean 95% confidence interval on the position, default max is 2.

*do\_avgsub* is a Boolean determining whether or not to subtract the mean of the off frames. Set to 1 to subtract the mean and to 0 to subtract the median. Default is 1.

*which\_gaussian* determines what functional form of Gaussian function the molecules will be fit to if using least-squares fitting (MLE fitting only fits symmetric Gaussian). Set to 1 to use a symmetric Gaussian. Set to 2 to use an asymmetric Gaussian (with axes along the row and column dimension). Set to 3 to use a freely rotating asymmetric Gaussian. Default is 1.

*Subtract\_then\_fit.m* outputs a .mat file called *moviename\_AccBGSUB\_fits.m* which contains the *fits* defined in the intro. The best way to debug the goodfit parameters is to watch the ViewFits movie. If you see a lot of guesses that you think actually are molecules not passing the goodfit checks (i.e., a lot of red circles instead of green circles around good looking molecules) then try relaxing the goodfit checks. Conversely, if you see a lot of false positives, then try tightening these parameters. The default parameters are the optimal parameters for a SNR of 2. It is far more time efficient to use the program *Change\_GoodFits* to change the goodfit parameters instead of refitting. Then simply remake the ViewFits movie and watch it to see how the change you implemented performed.

If *bgsb* was set to 0, then the program just does fitting and doesn't do any subtraction, (Maybe the function should be called *Subtract\_then-or\_fit.m*). In this case, the fits .mat file will just be called *moviename\_fits.m*.

## Tracking

*SMALLLABS\_main* calls a single particle tracking algorithm. The output tracks can be saved and analyzed separately. Because *SMALLLABS\_main* was designed with PAINT experiments in mind, the only use of this information incorporated into the main functionality is to filter out fits that aren't organized into a track, this is a useful way to ensure that fitting noise in the movie doesn't make it into the final result. The result is a logical vector for each fit called *trk\_fit*. Furthermore, the fits that were organized in tracks and are the first and last frame in a track are excluded from being set to 1 in *trk\_fit*. However, *SMALLLABS\_main* can be used for localization and tracking, the full tracking file can be saved, and there is an option to make a ViewFits movie with labels for visualizing tracking. Tracking occurs in a few programs the topmost wrapper program is *Track\_filter.m* which is called like:

```
Track_filter(fits_filename,append_vec,trackparams,savetracks);
```

*fits\_fname* the filename of the .mat file containing the *fits* array

*append\_vec* is a Boolean determining if the *trkflt* vector, and the *tracks* array will be appended to the fits .mat file

#### SMALLLABS\_main optional parameters

*trackparams* is a vector containing the tracking parameters. The elements in the vector and their default values are:

```
% minimum merit
trackparams(1)=0.01;
% Integration time (ms)
trackparams(2)=200;
% gamma
trackparams(3)=1;
% maximum step size
trackparams(4)=3;
% minimum track length
trackparams(5)=3;
% speed estimation window halfsize
trackparams(6)=1;
% time delay between consecutive frames (ms)
trackparams(7)=0;
```

*savetracks* is Boolean determining whether the output of the tracking file will be saved to a .mat file.

## ViewFits

The ViewFits movie is made with the function *ViewFits.m* which is called like:

```
ViewFits(movie_fname,fits_fname,circ_D,write_mov,autoscale_on,linewidth)
```

*movie\_fname* is the filename of the tiff stack movie to put the fit indicators on

*fits\_fname* is the filename of the .mat file containing the *fits* array

#### SMALLLABS\_main optional parameters

*orig\_movie* is a Boolean determining whether or not to use the original TIFF stack movie. Setting it to 0 will use the avgsub movie. Default is 1.

*circ\_D* is the diameter of the circles in the ViewFits movie. Default is *circ\_D* = *dfrlmsz* (which comes from the fits .mat file)

*write\_mov* is a Boolean that determines if the ViewFits movie will be written to an .avi movie. If set to 0 the program will go to debug mode allowing you to step through frame by frame to see the results.

*autoscale\_on* is a Boolean that determines if the movie grayscale will be set frame by frame. If set to 0 a handful of frames throughout the movie are used to set the grayscale for the entire movie. Default is 0.

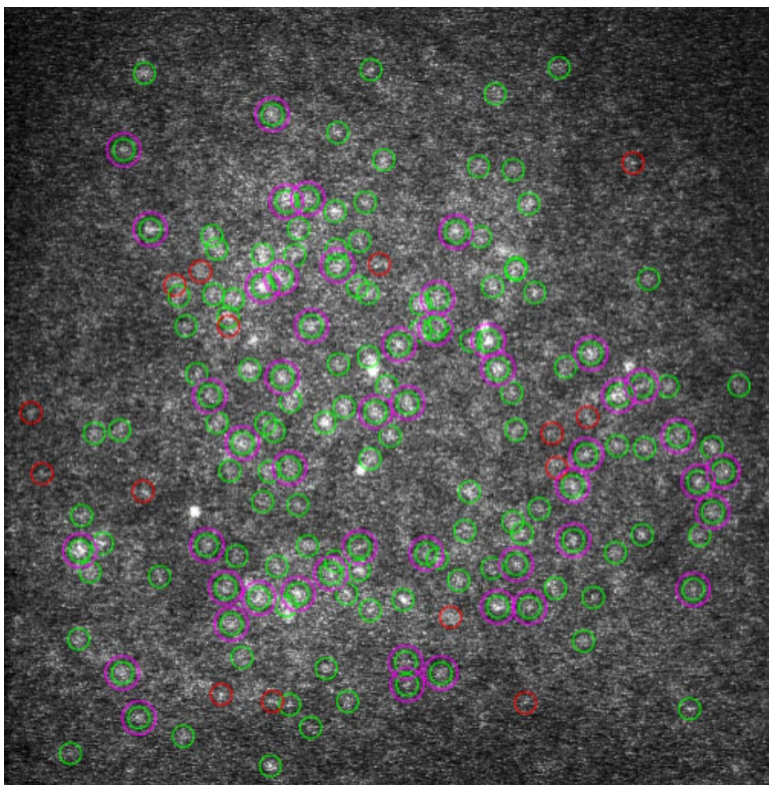
*linewidth* is the linewidth of the circles in the movie. Default is 1.

*trackingVF* is a Boolean that determines whether the tracking ViewFits program will be called instead. If set to 0 the ViewFits program described here is called. Default is 0.

In the ViewFits movie the color scheme is:

- green circles are good fits
- red circles are bad fits
- magenta circles are fits which passed the tracking filter

An example frame is:



## Fitting without background subtraction

*SMALLABS\_main* can also be used to do fitting without doing subtraction. This is accomplished by setting *bgsub* to 0. In this case the workflow is:

1. Guess molecule locations in the avgsub movie using *Guessing.m*



2. Fit using *Subtract\_then\_fit.m*, though no subtraction occurs

#### OPTIONAL STEPS

3. Track the fits to filter out fits which aren't organized into a track, and remove the first and last frame of the track using *Track\_filter.m*
4. Make a ViewFits movie to help check the fitting using *ViewFits.m*

Additionally, the filenames will be changed. Because the original movie is analyzed and not the avgsub movie, filenames won't have "\_avgsub" appended. Furthermore, the fits .mat file won't have *AccBGSUB* in its name.

## Coordinate System Definition

*SMALLLABS\_main* runs entirely in Matlab, and as such the coordinate system used is the most meaningful for Matlab arrays. Instead of returning x and y positions in some arbitrary coordinate system, *SMALLLABS\_main* 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 such  $(r,c) = (42.0597,153.9108)$ .

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 entitled *Image Coordinates in Matlab and ImageJ*.

## Speed and Memory Troubleshooting

*SMALLLABS\_main* 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 if that is needed.

**Use smaller movies.** Obviously using smaller movies will require less memory and time to analyze them. You can use the program *ROI\_picker* to pick a smaller region of interest (ROI) within your original movies.

**Fit fewer molecules with least squares.** The slowest step in *SMALLLABS\_main* is fitting. If you reduce the number of guesses, then the program will spend less time trying to fit those guesses. 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 is much faster than MLE fitting, setting *MLE\_fit* to 0 accomplishes this.

**Guess frame by frame.** In the guessing step, if *pctile\_frame* is set to 0 (the default), *bpthrsh* is calculated from all of the frames. This means that the entire movie will be loaded into Matlab, so if you have a large

movie this will take up a lot of memory. If that is a problem, set *pctile\_frame* to 1, so that only individual frames are loaded in this step.

**Don't make a ViewFits movie.** The optional step to make a ViewFits movie is quite slow. Because *SMALLLABS\_main* does all of the analysis before getting to this step and saves all of the results, you can in principle start analyzing the fit results while the program is still making ViewFits movies. However if that isn't convenient for you, just set *makeViewFits* to 0 and make the ViewFits movies only as needed.

## Additional Programs Provided

A number of additional programs which are not needed for operation in *SMALLLABS\_main* are provided as they can be very helpful. The following is a short description of these programs; see the comments at the beginning of each program for more details.

*Change\_GoodFits.m* is a function which can be run after fitting to change the goodfit parameters (*maxerr* and *stdtol*) and also change *dfrlmsz* and consequently 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.

*ROI\_picker.m* is a function which allows you to pick an ROI in a series of movies and write a new TIFF stack 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.

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.

## The modularity of this code

*SMALLLABS\_main* is written as modular code. It was designed so that different parts of the code can be easily modified or even completely replaced. Throughout the code I've tried to copiously comment, and in particular explain the inputs and outputs of the various steps and functions. This should help make any modifications or replacements simpler to implement.

In particular, I anticipate users may wish to utilize a specific guessing (detection) method or specific fitting method. For example if a non-Gaussian PSF is used then the provided fitting functions won't work and the guessing algorithm may not work. By simply modifying those two steps *SMALLLABS\_main* 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, which is as a .mat file

using the same naming convention, and with an array called *guesses* that has columns 1. frame #, 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 in *Subtract\_then\_fit.m*. In this new section fit the same array as in the other sections, called *data*. Then enter the fit results in to the *fits* array using the column definitions described in the introduction of this guide. Note that the goodfit *if statement* in the following section may need to be modified to be appropriate for the new fitting method.