User Manual

This manual provides a detailed guide for the use of two programs written in the MATLAB programming language that facilitate counting, length measurements, and quantification of bundling of actin filaments visualized in fluorescence micrographs. Installation of the following MATLAB toolboxes is required to run these programs: (1) Image Acquisition Toolbox, (2) Image Processing Toolbox, (3) Signal Processing Toolbox, and (4) Curve Fitting Toolbox.

Filament_Length_Quantification

Purpose

This program measures the lengths of filamentous objects in a single snapshot. It can also be used to quantify the number of filaments or bundles in the image.

General Comments/Tips

Red-green colorblind people may have difficulty using this program. In that case the user can change the colors used to highlight filaments in the code. To do so, change the `plot` color input argument from `'r.'` and `'g.'` (red and green) to `'c.'`, `'m.'`, or `'y.'` (for cyan, magenta, or yellow).

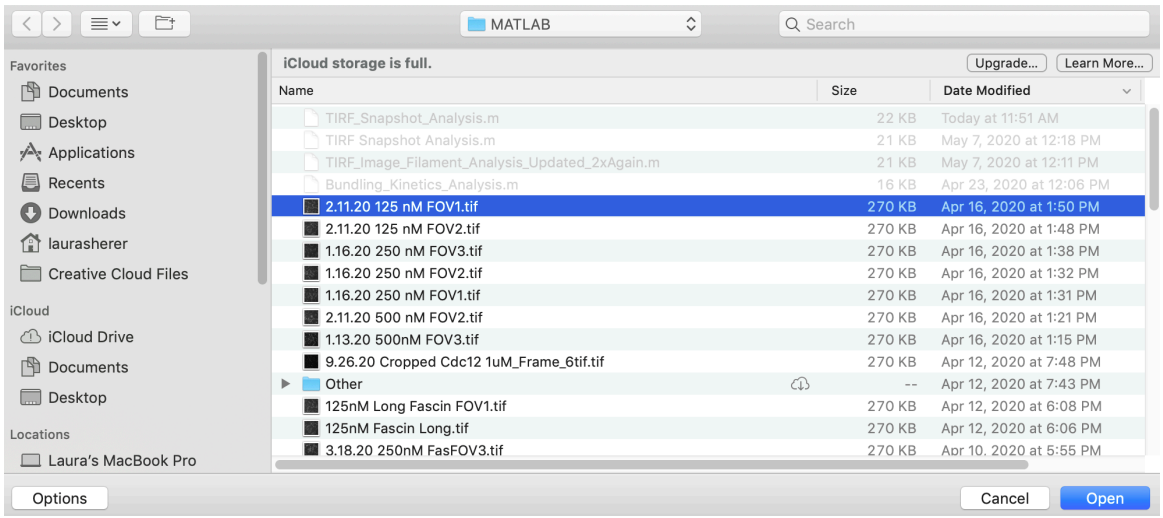
When a user clicks close to an object, the program will register it as clicking on that object. This aspect of the program was intentional because it can be hard to click on a narrow line. If the user's click is not near any object, the program will ask the user to click again.

Any mistake made during the error correction process cannot be undone, so the analysis should be restarted in such a case.

If the program runs for a long time (more than 1-2 minutes), the user should stop running the program and change the parameters.

Instructions

1. Run the program. A file finder window will automatically appear. **Select one or more TIRF micrographs to analyze.** Hold down the command key ("control" in Windows) while clicking to select multiple images. Only micrographs in TIF format can be selected. **NOTE:** The height and width of the image in pixels must be the same, and they must be divisible by the variable called section_pixel_width.

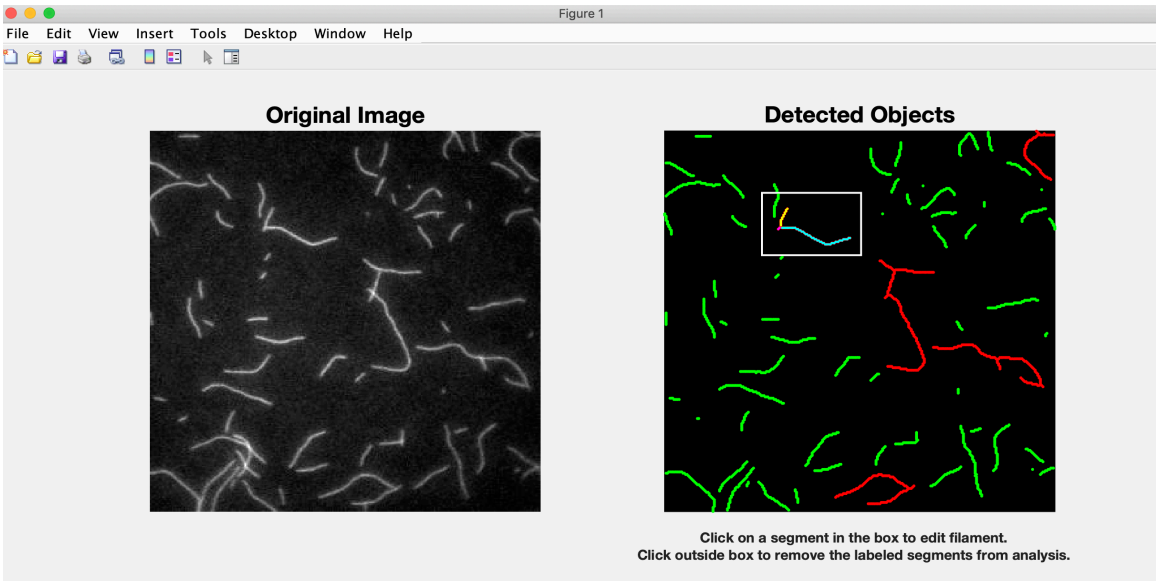


2. **Select parameters.** This step sets the background_base and noise_filter_base variables. If an image is crowded, the recommended selection is Filter 3. For less crowded images, the recommended selection is Filter 1. Filter 2 is an intermediate setting. If object detection is unsatisfactory with the selected filter, restart the program and choose a different filter. The user can also manually change the parameters by adjusting the noise_filter_base and background_base variables in the code. If the user selects multiple images, the same filter will be applied to each image.

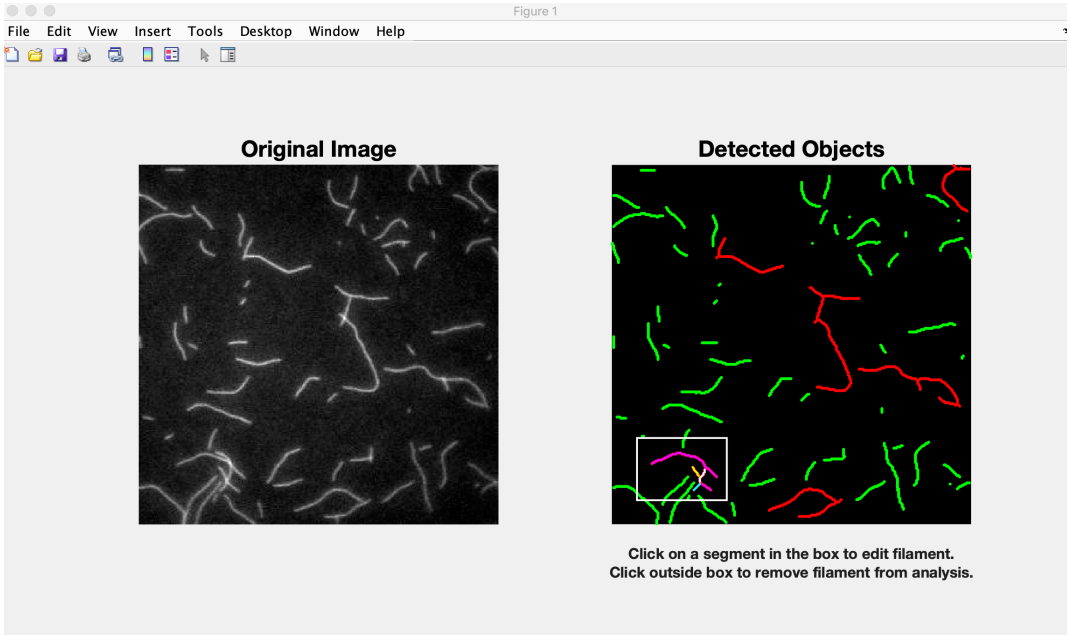
3. Next, the original image and an analysis image are displayed. Filaments that have two endpoints and no “branchpoints” (i.e., overlapping segments) are automatically labeled in green. Filaments with topological anomalies are automatically labeled in red. **Only red filaments will undergo manual error correction by the user in the following steps.**

If the user wishes to include additional objects in error correction or omit them from it**, they can click filaments to switch the labeling from red to green (or vice versa)**. The user presses the “return” key (“enter” in Windows) when finished.

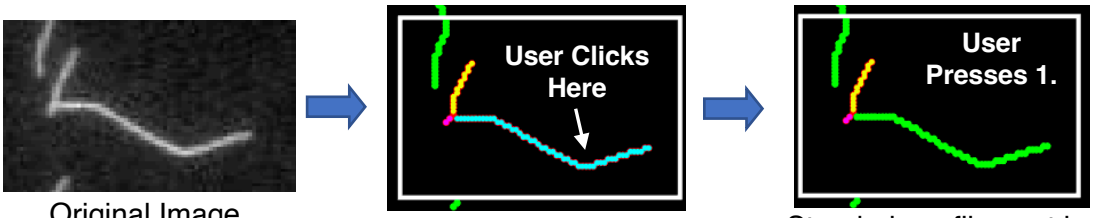
4. Steps 4 and 5 are part of the manual error correction process. During error correction, a white box highlights one object at a time. **To completely remove a misidentified object from analysis, click anywhere outside the white box on the micrograph.** Complete removal of an object is a good option if the filament is mostly outside the field of view or if it contains more than two unresolved segments.



5. **To correct the error instead of removing the whole object from analysis, click on a labeled filament segment within the box.** The selected segment will turn green. **The user will be prompted to select a key to specify how to resolve the error** (described below with an example). When there are no more segments to resolve within the white box, the program will automatically move onto the next filament. **If the user has corrected a few segments but wishes to remove the remaining unspecified segments from analysis, they can click away from the box.** (This option is useful when the remaining segments are very small.) The program will cycle through each of the objects that are highlighted in red until they are all resolved.

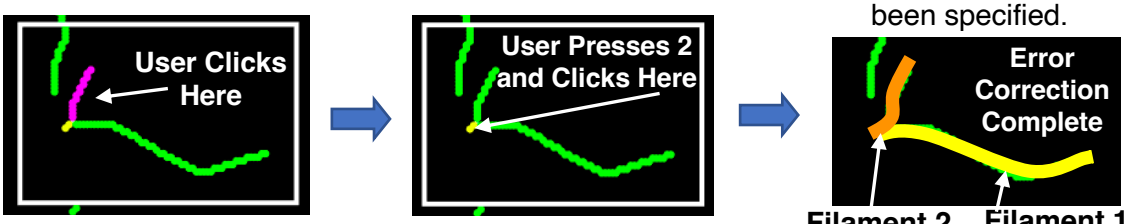


Zoomed-in view of the manual error correction process:



Original Image

Stand-alone filament has been specified.



User must specify remaining segments (or click outside the box to move on).

User specifies that these two segments go together.

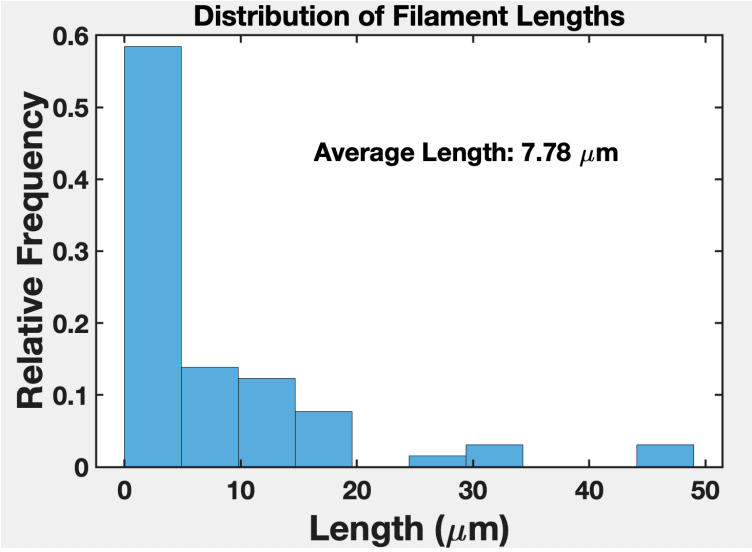
Press key to specify edit.
Esc: Segment should be removed from analysis.
1: Segment is a stand-alone filament
2: Segment is part of another filament.

Note: Pressing 2 only allows for the combination of two segments into a single filament.

Note:
Segments removed from analysis will be removed from the image altogether. Resolved stand-alone and multi-segment filaments will remain green and appear unchanged. The yellow and orange lines in the above example are intended only to illustrate which segments are counted as individual

6. After error correction is complete, **the program generates a histogram of filament lengths.** If the user has selected multiple images for analysis, the data from all of the images will be combined into a single histogram. The length calculation is based on the following conversion: 1 pixel = 0.2667 microns. This conversion factor was determined based on a camera with 16 μm x 16 μm pixels and an objective that provides 60x magnification. This conversion factor can be modified according to the user's camera and objective by editing line 311, as shown below:

```
311 -    lengths_micrometers = lengths_pixels * 0.2667;
```



Bundle_Assembly_Movie_Analysis

Purpose

This program measures the fraction of filamentous actin that is bundled over time. The user can directly infer the length of the delay prior to the onset of bundling from the output graph. By applying exponential fits to the analyzed data, the user can also obtain bundle assembly rate constants.

General Comments

Before you run this program, you should first obtain the "photobleaching factor". This enables the Bundle_Assembly_Movie_Analysis program to account for changes in fluorescence intensity that may arise from photobleaching over the length of the reaction. The photobleaching factor can be obtained using the Photobleaching_Correction program as described in the following section.

The fluorescence intensities used by the program during bundle quantification correspond to fluorescence intensities subtracted by the surrounding background fluorescence. Segmenting the micrograph into a grid as described below enables local background subtraction. This increases the accuracy of bundle detection and allows for more accurate comparisons of bundle assembly across different reactions.

The fraction of filamentous actin bundled over time is calculated by the following equation:

$$Fraction\ Bundled = \frac{F_{Bundles}}{F_{Total}}$$

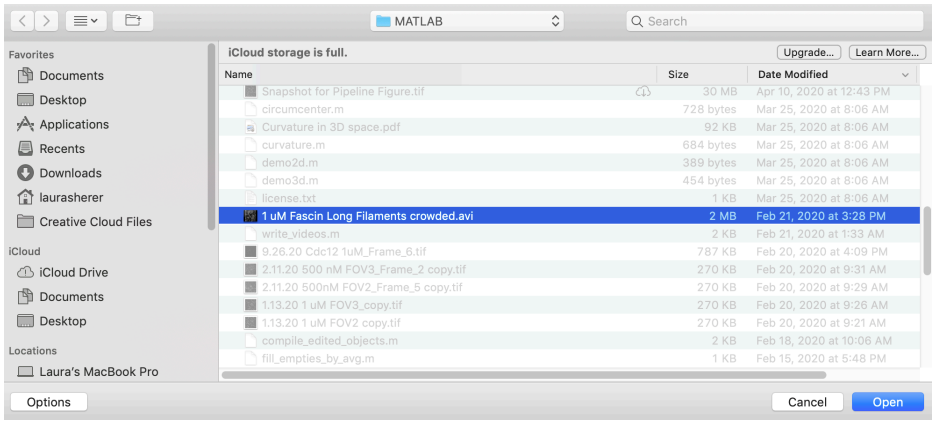
where $F_{Bundles}$ is the fluorescence intensity of pixels above the threshold fluorescence for bundles (set by the program) and F_{Total} is the total fluorescence intensity of all pixels. This metric was chosen as it accounts for the expansion of bundles (i.e., incorporation of additional filaments into a pre-existing bundle). It also accounts for differences in the total amount of filamentous actin at different timepoints in the same reaction and across different reactions.

To enable direct comparison of different bundle assembly movies, the dynamic range of the fluorescence intensity during the bundle assembly process should be similar. For example, if the fluorescence intensity is saturated for a 4-filament bundle in one movie, the fluorescence should be saturated at 4-filament bundles for other movies as well.

The program analyzes each frame in a movie (which is often > 100 frames), so analysis will take ~20 seconds to complete. If the analysis is taking more time than that, we recommend modifying the parameters and trying again.

Instructions

1. Run the program. **Select a single bundle assembly movie in avi format from the file finder window.** The first frame should show only single (i.e., not bundled) filaments so that the program can establish a fluorescence intensity baseline and determine a suitable bundle intensity threshold. The image width and height in pixels should be divisible by the variable called split_number.



2. **Set the parameters in the dialog box** (see screenshot below)**.**

   The “Time Interval between Snapshots” should be indicated in seconds.

   The Micron/Pixel Ratio is determined by the properties of the camera and the magnification used to collect the image.

   The “Noise Filter Base” and “Background Base” correspond to the width of the Gaussian blurs that are applied to the image for noise and background filtering. These values are determined by multiplying the desired standard deviation of each function by a factor of seven, which accounts for more than 99% of the pixels that may contribute to the signal in each region of the micrograph. For best results, the width of the Gaussian blur used for the background filter should be larger than that of the noise filter. We recommend using standard deviations of 1 and 9 pixels for the Noise and Background filters, which correspond to base widths of 7 and 63, respectively. These values can be adjusted individually and iteratively to improve the accuracy of filament resolution and noise filtering.

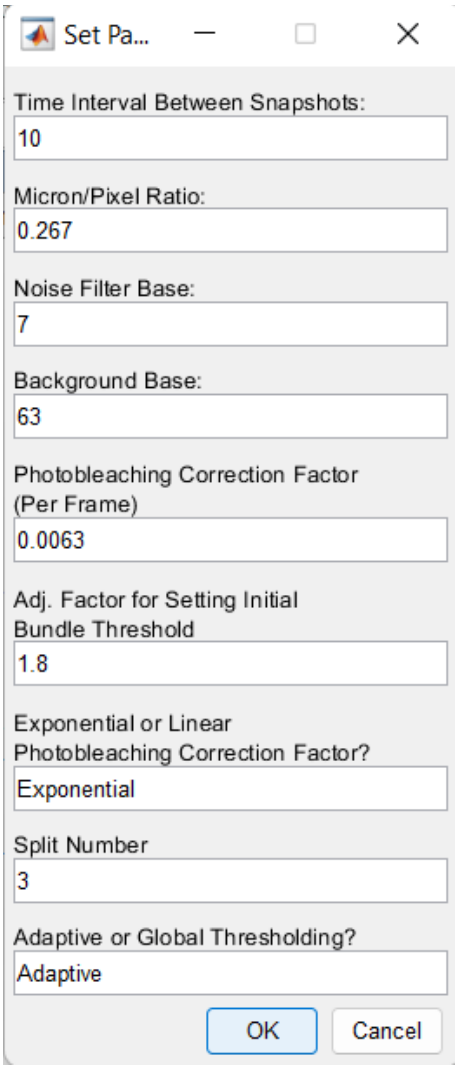
The “Adj Factor for Setting the Initial Bundle Threshold” (called start_corr in the code) is the number that will be used in the equation below. This number sets the value of the fluorescence threshold above which a filamentous object will be detected as bundled. This value is calculated relative to the average fluorescence intensity of individual, non-bundled filaments. We recommend beginning with a value of 1.8, which dictates that any pixel with a fluorescence signal that is greater than or equal to 1.8 times the average intensity of a single filament will be recorded as “bundled”. **If the program is not detecting bundles accurately, we recommend changing this parameter first.**

```
thresh_intensity_ROI(row_idx) = mean(pixel_intensities) + start_corr * std(pixel_intensities);
```

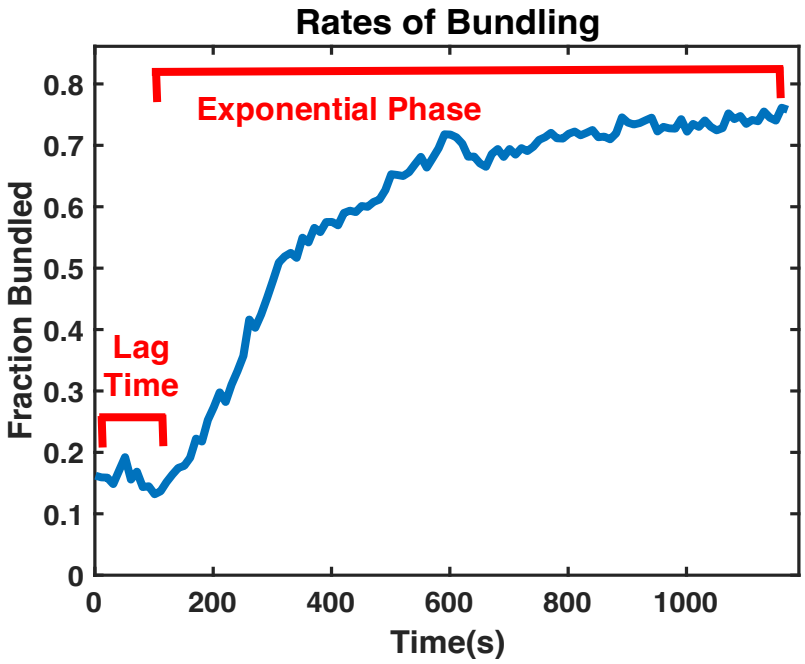
The photobleaching correction factor is obtained using the program described in the following section. The user is also asked to specify whether a linear or an exponential fit was applied to determine the photobleaching correction factor. This can be done by typing “Linear” or “Exponential”.

The “Split Number” specifies how the image will be segmented into a grid to account for uneven illumination. A split number of 2 means the image will be divided into 2 sections in both the x and y directions (i.e., four regions total). A bundle threshold will be calculated for each section of the grid. It is recommended that users change the “Split Number” variable if there is uneven illumination that causes different areas of the image to have very different fluorescence intensities.

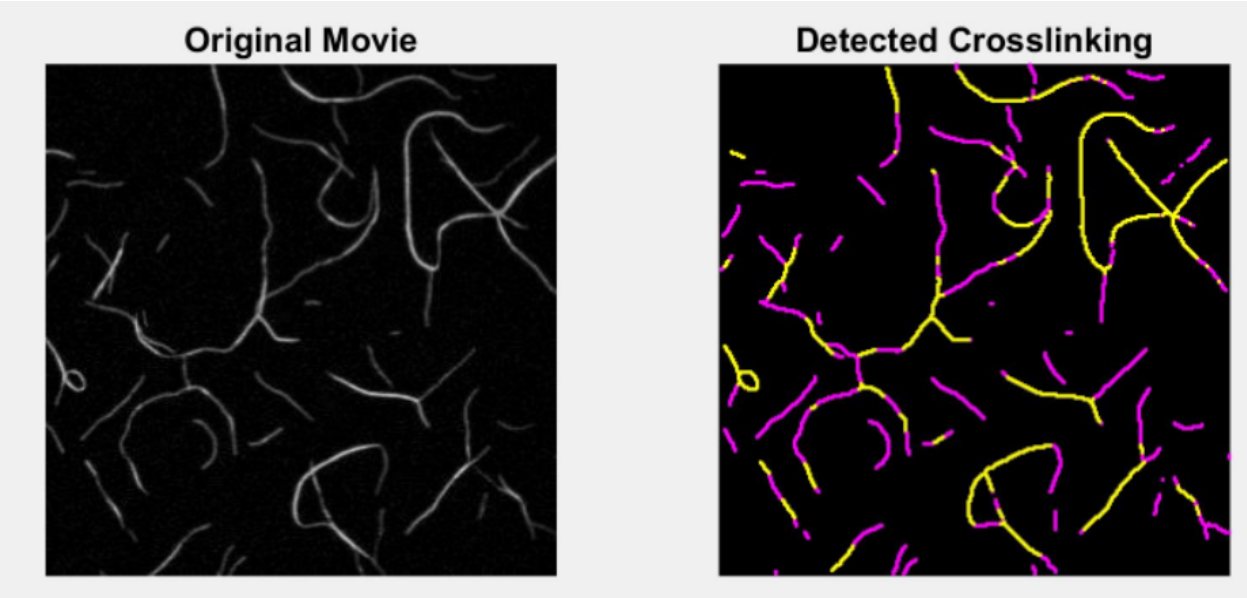
“Adaptive or Global Thresholding?” refers to the thresholding process that is involved in image binarization. Global thresholding is used to determine the threshold value based on the global distribution of pixel intensities across the entire image. Adaptive thresholding is used to calculate different threshold values for smaller regions of the image. We recommend using adaptive thresholding for movies with large bundles, which can produce significant variability in the pixel intensities across fluorescent micrographs. (The program accepts both “Adaptive” or “adaptive” with any number of spaces.)



3. After the analysis is performed, a graph depicting the fraction of actin that is bundled over time will appear. The user can infer the length of the delay prior to the onset of bundling (i.e., lag time) directly from the graph. The user can also obtain bundle assembly rate constants by fitting a curve to the exponential phase of the graph with first-order integrated rate law with a single or double exponential.



4. **The program also generates an analysis movie** (as shown in the image below)**.** The movie is automatically saved in the working directory in the form of “movie title_analysis”. Important parameters and outputs are also saved as “movie title_variables” or “movie title_fraction_bundled”. Single filaments are labeled in magenta. Bundled filaments are labeled in yellow. If the analysis is unsatisfactory, we recommend adjusting the variable “Adj. Factor for Setting the Initial Bundle Threshold” first in the dialog box as described in Step 2.



How to Obtain the Photobleaching Correction Factor

Use the following program, which is located in the Bundling Analysis folder:
Photobleaching_Correction

Purpose

This program quantifies the change in the average fluorescence intensity of single filaments in a bundling movie over time. This program can be used to obtain a photobleaching correction factor to be input into the movie analysis program described above.

General Comments and Tips

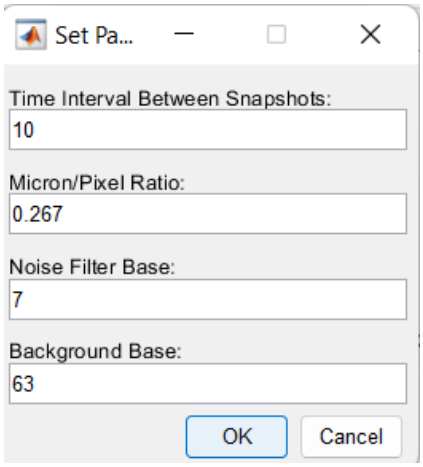
Background subtraction in ImageJ is recommended prior to using this program.

Users should have the Curve Fitting Toolbox installed to run this program.

Instructions

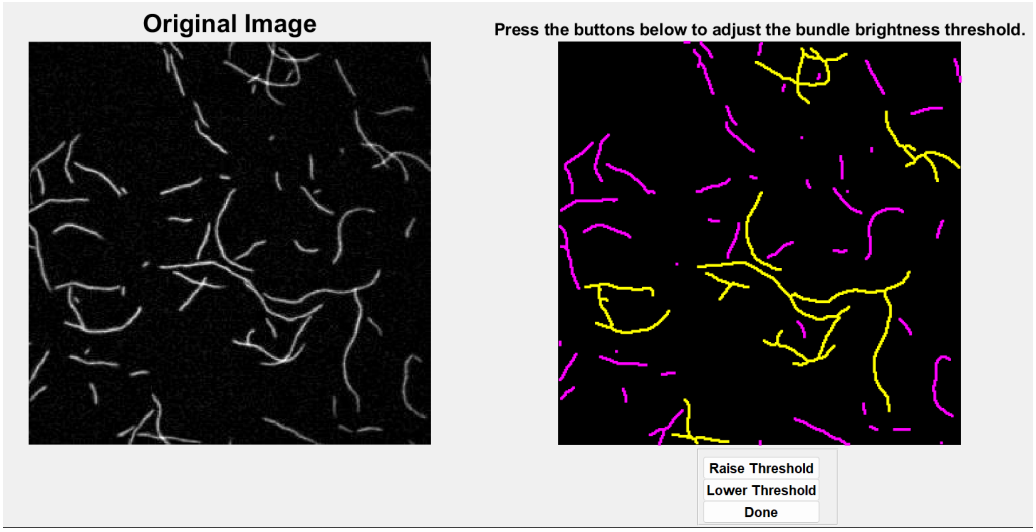
1. Run the program. A file finder window will appear. Select the movie to be analyzed. It should be in AVI format.

2. Set parameters. “Time Interval Between Snapshots” is the time between successive frames in the movie and has units of seconds. The Micron/Pixel Ratio is determined by the properties of the camera and the magnification used to collect the image. The Noise Filter and Background Bases correspond to the width of the Gaussian blurs that are applied to the image for noise and background filtering, as described above.



3. Click on the buttons below the analysis image to adjust the bundle threshold so that only filaments containing overlapping or bundled regions are shown in yellow. The intensities of the filaments highlighted in magenta will be pooled together to calculate an average fluorescence intensity for single filament regions. It is not necessary for every single filament region to be highlighted in magenta, but you should have enough to get a representative average intensity value across the whole image. The default adjustment with every button press is quite small, so you may need to press the button multiple times before you see changes in the image. (If you would like to change the step size with every button click, change the value = 5 in the set_bundle_threshold_photobleaching below.) When you're done, press the Done button.

```
if lower_thresh.Value == 1;
   brightness_threshold = brightness_threshold - 5;
   delete(lower_thresh);
   lower_thresh = uicontrol(group,'Style','togglebutton','Position',[7 30 150 30],'String', 'Lower Threshold','FontSize',13,'FontWeight','bold',...
   'Callback', 'uiresume');
elseif higher_thresh.Value == 1;
    brightness_threshold = brightness_threshold + 5;
    delete(higher_thresh);
    higher_thresh = uicontrol(group,'Style','togglebutton','Position',[7 58 150 30],'String', 'Raise Threshold','FontSize',13,'FontWeight','bold',...
    'Callback','uiresume');
end
```



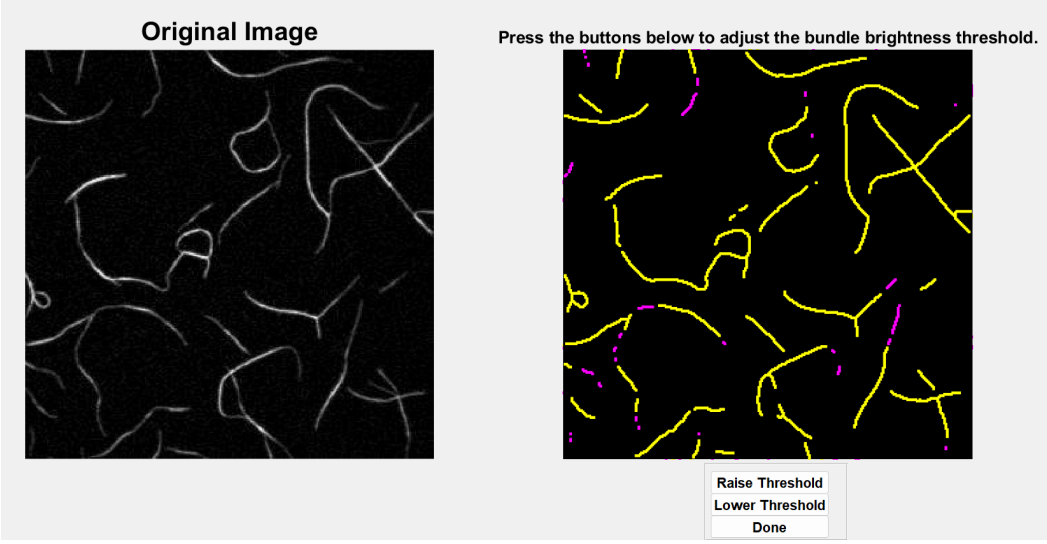
4. Next, the program will skip forward 20 frames and display the next frame. Repeat the process as outlined above. This will continue until the end of the movie is reached. If you would like to collect more datapoints for the fits, increase the frame_interval variable. If you would like to collect fewer data points, decrease this variable.

```
frame_interval = 20;
% the above line means single filament intensities will be
%measured every blank frames
```

Because this program uses a global thresholding algorithm, **the appearance of larger bundles will often lead to fragmentation of the filaments and bundles** (shown below). This is acceptable and often preferable so that single filament regions can be isolated from bundled regions. If you find that the global thresholding algorithm is not detecting enough single filament regions, comment out the imbinarize(norm_image, threshold_mask) line and uncomment the line below it to use an adaptive thresholding algorithm. However, be careful that the adaptive thresholding doesn't lead to the program picking up background noise, which can affect the average intensities measured for single filaments.

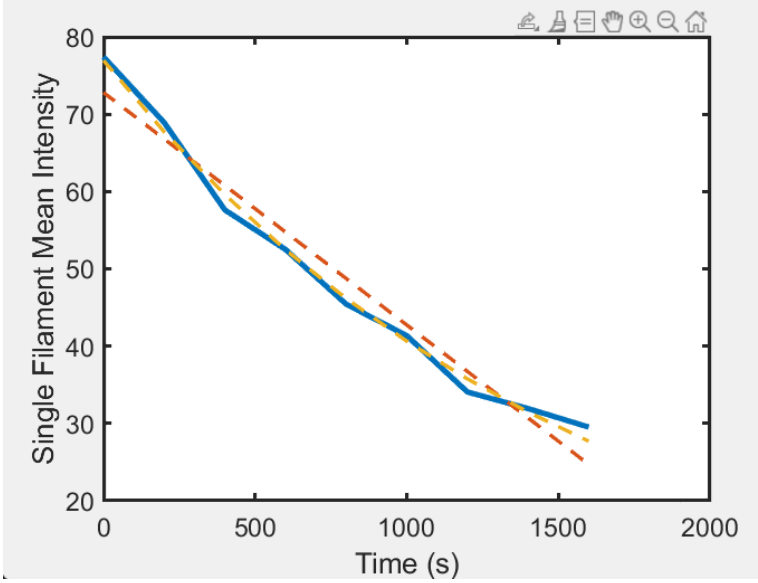
```
BW_image =imbinarize(norm_image, threshold_mask);
%BW_image = imbinarize(norm_image,'adaptive','ForegroundPolarity','dark','Sensitivity',1); %convert to black and white
```

We find that the global thresholding algorithm is usually adequate for the purposes of calculating the photobleaching correction factor. For example, although the analysis below doesn't identify all filaments correctly, it is useful for obtaining the pixel intensity values of single filaments (magenta) in different regions of the image.



5. Once the user has set the threshold for each frame, the program will generate a graph depicting the average intensity value for single filaments over time, along with linear (dark orange) and single-exponential (light orange) fits to the data (shown below). The program will display the fit information (slope, y-intercept for linear and a, b for exponential) and the r squared values in the command window. It will also save the graph, input parameters, and output variables in your MATLAB folder. The files will be called movie_name_PBC_graph and movie_name_PBC_variables.

We typically use the exponential fit to generate the photobleaching correction factor because it is often the best fit. **Please note:** The exponential fit parameter defined as "b" has units of fluorescence pixel intensity per second. When it is entered into the bundle analysis program, it must be converted to fluorescence pixel intensity per frame. This can be done by multiplying "b" by the time interval at which frames are collected. This value also should not be negative when entered into the bundle assembly program. An example is shown below.




```
General model Exp1:
y(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
  a =       76.95   (74.58, 79.31)
  b =  -0.0006385   (-0.0006846, -0.0005924)
```

This is the value to enter into the bundle assembly program. (Photobleaching Correction Factor Per Frame)

Our example movie collects one image every 10 seconds. So, to set the Photobleaching Correction Factor in the bundling program, we would enter: 0.0006385 x 10 = 0.006385

List of Functions (For best results, make sure these are in your MATLAB folder.)

Functions created for our programs

closest_to_click:

returns the index of the object closest to a user click (unless the click is more than 15 pixels away from any object in either the x or y direction)

highlight_branches:

creates a box around the filament of interest, finds the "branches" or segments that are part of the filament of interest, and labels those segments of interest with different colors.

pixel_struct_to_mat:

reformats a structure field containing (x, y) pixel list coordinates into a matrix (cuts down on loops needed later). Each object has its own row of x and y values. Zeros are at the end of smaller objects to fill up the matrix. The matrix format is:

$$\begin{matrix} x1 & y1 & x2 & y2 & x3 & y3 \\ x1 & y1 & x2 & y2 & 0 & 0 \\ x1 & y1 & 0 & 0 & 0 & 0 \end{matrix}$$

identify_problem_filaments:

returns the index number of detected objects that contain topological anomalies like a "branchpoint" or more than 2 endpoints.

calculate_background:

finds the average background fluorescence intensity for each section of the image grid

resolve_mutlisegment_fil:

used for manually resolving a filament with two segments. It enables the user to select another segment of a filament and determines the perimeter of the two clicked segments together.

compile_edited_objects:

adds manually edited filaments pixel coordinates into pixel matrix of previously detected objects

fill_empties_by_avg:

fills empty sections of the thresh_intensity_ROI array with the average pixel intensity obtained from the neighboring grid sections of the image

write_videos

displays original bundle assembly movie side-by-side with the bundle analysis (bundles shown in yellow) and saves each frame of the figure as a movie

set_bundle_threshold

generates the buttons that allow for adjusting the bundle threshold in the bundling program and displays filaments/bundles above the threshold in yellow

set_bundle_threshold_photobleaching

generates the buttons that allow for adjusting the bundle threshold in the photobleaching correction program and displays filaments/bundles above the threshold in yellow

Functions Made by Others in the MATLAB Community (Cited Sources Below)

myginput:

gets the coordinates of a user click and allows for customization of the pointer. (The MATLAB ginput function is similar but uses crosshairs instead of an arrow as the pointer. The crosshairs were harder to see against the black images.) Submitted onto MathWorks File Exchange by Frederic Moisey. (https://www.mathworks.com/matlabcentral/fileexchange/12770-myginput)

getkey:

waits for a key press from the user and returns the ASCII code to specify which key was pressed. Submitted onto Mathworks File Exchange by Jos van der Geest. (https://www.mathworks.com/matlabcentral/fileexchange/7465-getkey)

Uncommon Built-In MATLAB Functions (paraphrased from MATLAB documentation)

uigetfile: allows user to select a file from a finder window

questdlg: creates a question dialog box and returns the user's answers

bwskel: reduces objects in a binary image to their 1-pixel wide centerlines, preserves their topology

regionprops: measures specified properties of objects from an image and returns them as a structure

bwmorph(skeletonized_image, 'branchpoints'): returns the branchpoint pixel coordinates of the skeletonized_image