The ITRIA Interactive Tracking roGFP Image Analysis MATLAB tool user guide

Avia Mizrachi

29.10.2018

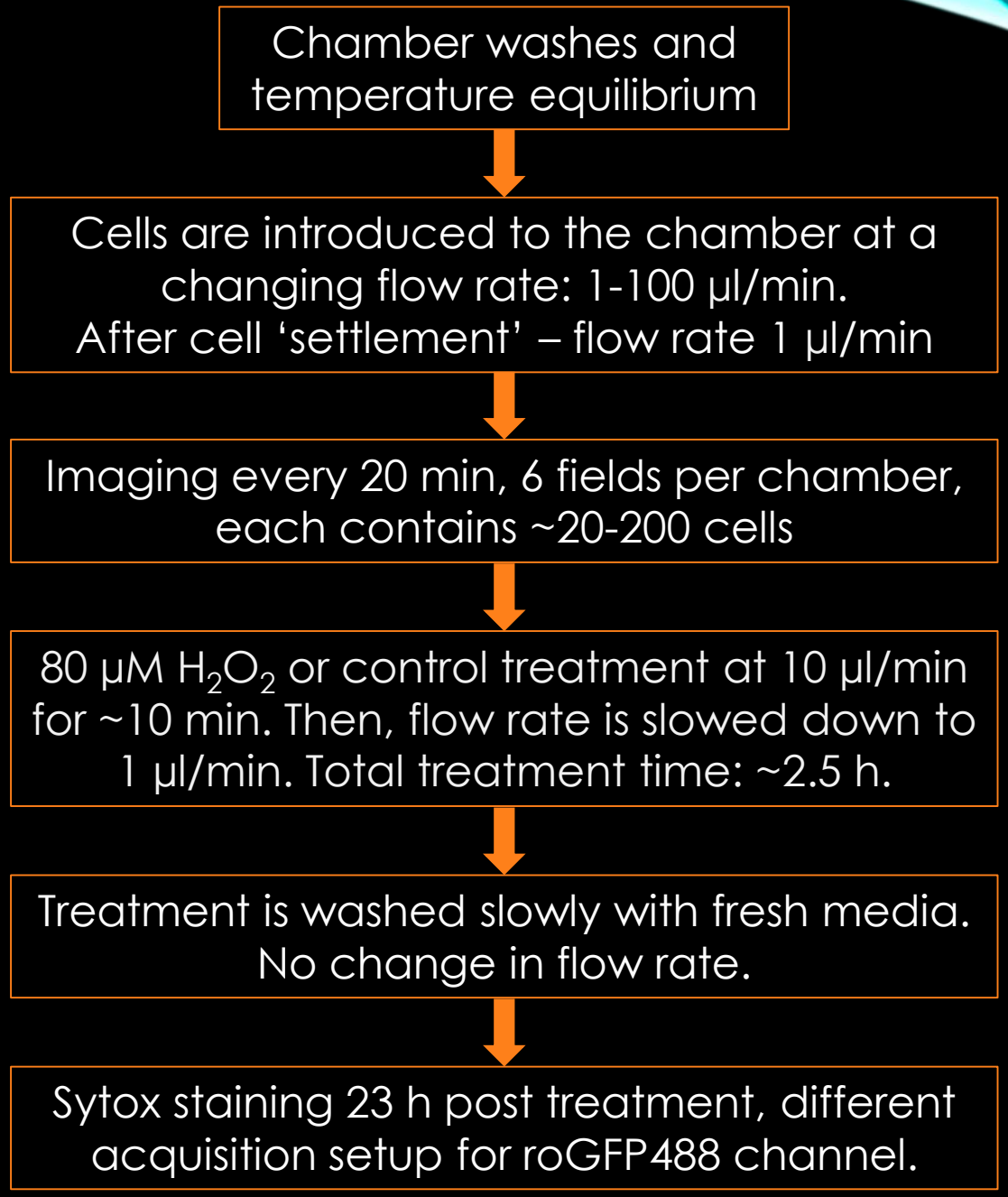
Experimental setup:
A microfluidics system for live-imaging

Cells expressing chl-roGFP and WT were imaged for 24 hours following treatment of 80 µM H_2O_2 or fresh media (control) using a microfluidics chamber.

To quantify cell death the same cells were stained with **Sytox Green** 22.5 hours following treatment.

Image acquisition order:
1. roGFP405 (i405)
2. roGFP488 (i488 or Sytox green)
3. Chlorophyll (Chl)
4. Bright field (BF)

***No imaging during the night!**



Itria: goals

- Measure roGFP oxidation over time from the same cells:
  - Cell identification
  - Tracking
  - roGFP calculations
  - Optional: autofluorescence measurement (especially in WT). → work in progress
- Link early oxidation patterns with cell fate:
  - Identify Sytox stain
  - Link Sytox stain to the same cells that were tracked over time.
- Data analysis:
  - Oxidation dynamics
  - cell fate
  - Optional: auto-fluorescence

The Itria: step by step

1. Loading imaging and analysis parameters
2. Open and concatenate the stacks + get time-points data
3. Pre-processing:
   - Image stabilization
   - Bright field (BF) band-pass filtering
4. Processing:
   - roGFP405: bg subtraction + thresholding
   - roGFP488: bg subtraction + thresholding
   - Chl: bg subtraction + thresholding
5. Co-localization mask: 405 & 488
6. roGFP expression + mask: 405 * 488
7. roGFP ratio: 405/488
8. oxD calculation
9. Auto-fluorescence (AF) ratio: 405/chl
10. Sytox: time-point, mask.
11. Cell segmentation
12. Cell tracking
13. Track linking: link missing frames
14. Track trimming: remove short tracks
15. Data analysis & plots
    - Oxidation over time + cell fate
    - Track trajectories

1.Loading imaging and analysis parameters

At the beginning of the script the user can choose to load variables and parameters that were used previously, so that all the stacks will be analyzed the same.

If not, new variables are initiated based on predefined values in the "varsFunc.m" file.

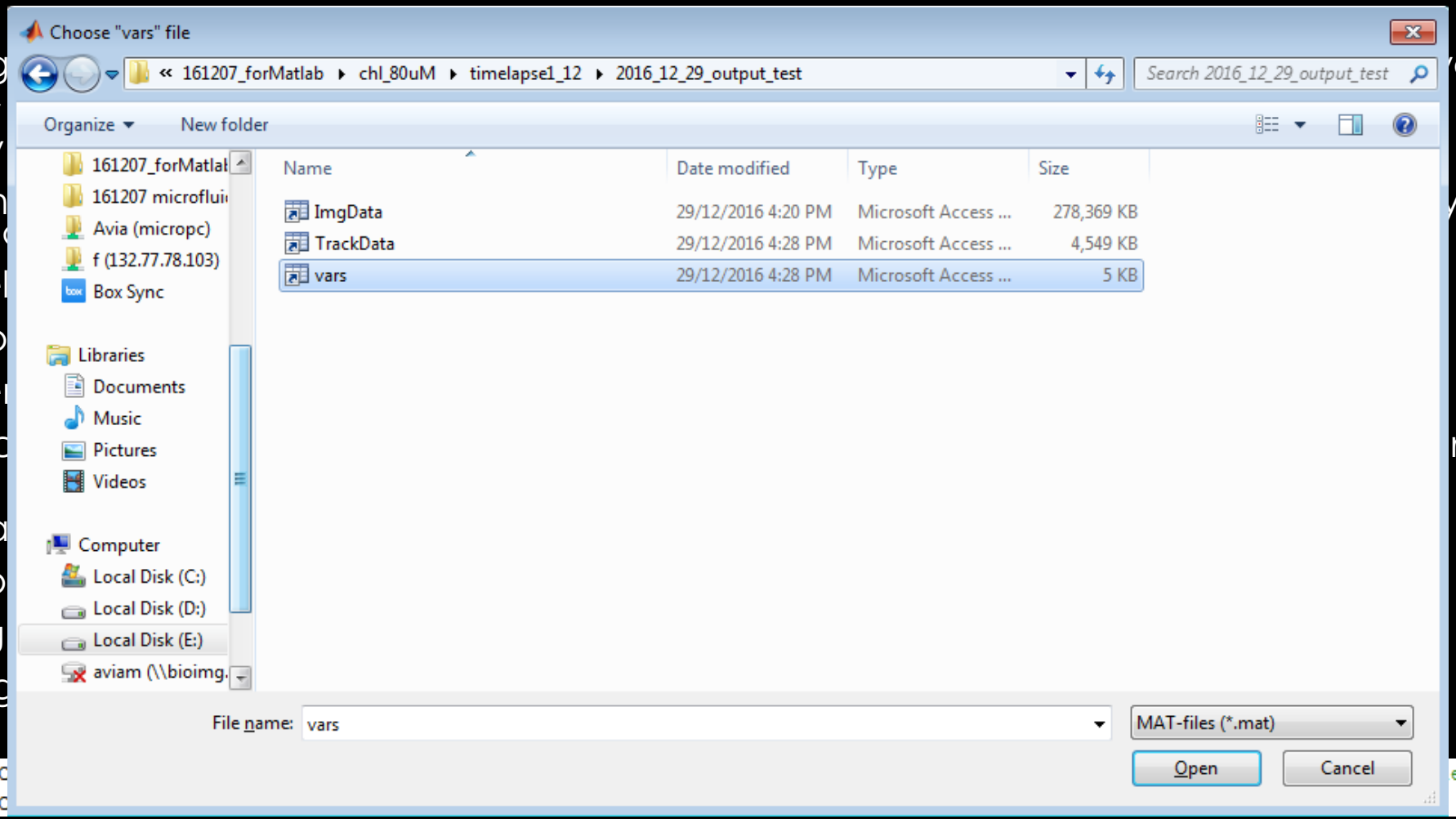
Many of the variables can be changed during later stage of the analysis, but some are very important and can't be changed later:

- Channel acquisition order
- Image bit depth (14-bit, depends on aquisition)
- Treatments times + names
- convFactor – conversion from pixels to actual length. Not critical at the moment, important ion order to calculate actual speed.
- frameRate – also not that critical at the moment, important for speed calculations.
- Oxidation parameters (based on extreme treatments): max oxidation, max reduction.
- Tracking parameters
- saveFlag: whether to save files along the way

```
varsChoice='new'; %'new'; %options: 'new' or 'load'. New to create the vars variable, load to load previous variables.
timeChioce='new'; %Whether to load aquisition time parameters or create new ones. options: 'new' or 'load'.
```

1.Loading imaging and analysis parameters

At the beg ere used
previously,
If not, new
Many of th y
important
- Channel
- Image b
- Treatme
- convFac nt ion
order to
- frameRa
- Oxidatio
- Tracking
- saveFlag




```
varsC
timeC
```

```
es.
```

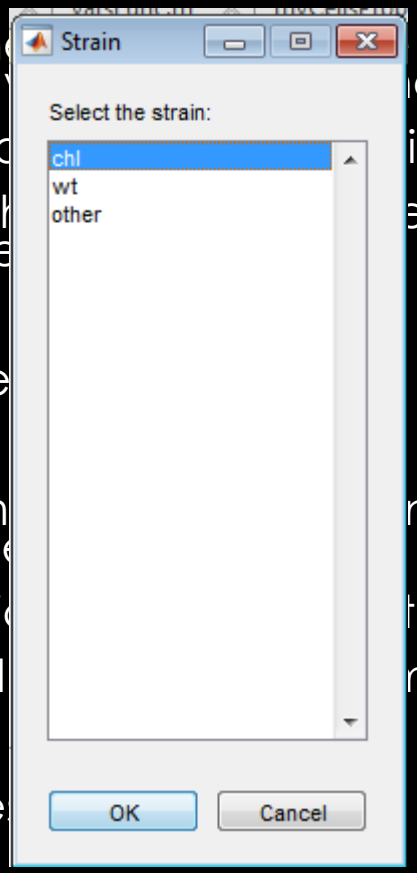
1.Loading imaging and analysis parameters

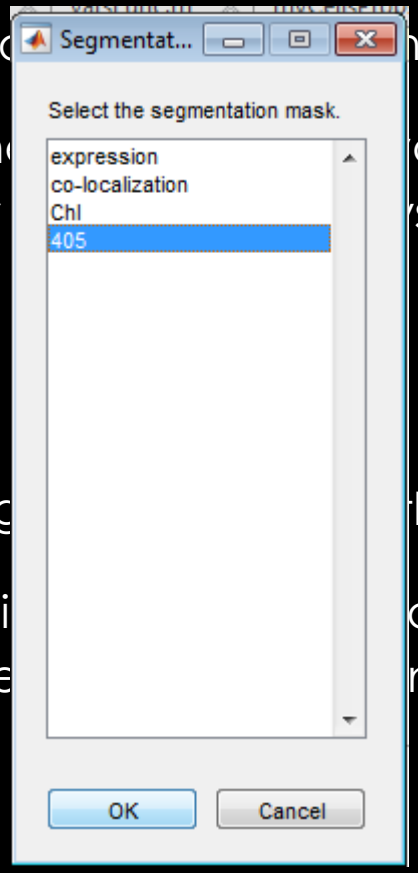
At the beginning of the script th… t… nd parameters that were used previously, so that all the stacks …e

If not, new variables are initiated … in… arsFunc.m" file.

Many of the variables can be ch… er … sis, but some are very important and can't be change…

- Channel acquisition order
- Image bit depth (14-bit, depe…
- Treatments times + names
- convFactor – conversion from… ng… he moment, important ion order to calculate actual spee…
- frameRate – also not that criti… t, i… d calculations.
- Oxidation parameters (based… ne… n, max reduction.
- Tracking parameters
- saveFlag: whether to save file…






```
varsChoice='new'; %'new'; %options: 'new' or 'load'. New to create the vars variable, load to load previous variables.
timeChioce='new'; %Whether to load aquisition time parameters or create new ones. options: 'new' or 'load'.
```

2. Open and concatenate the stacks + get time-points data

The script opens the original ".vsi" files that are saved during imaging, so there's no need for exporting or file conversion.

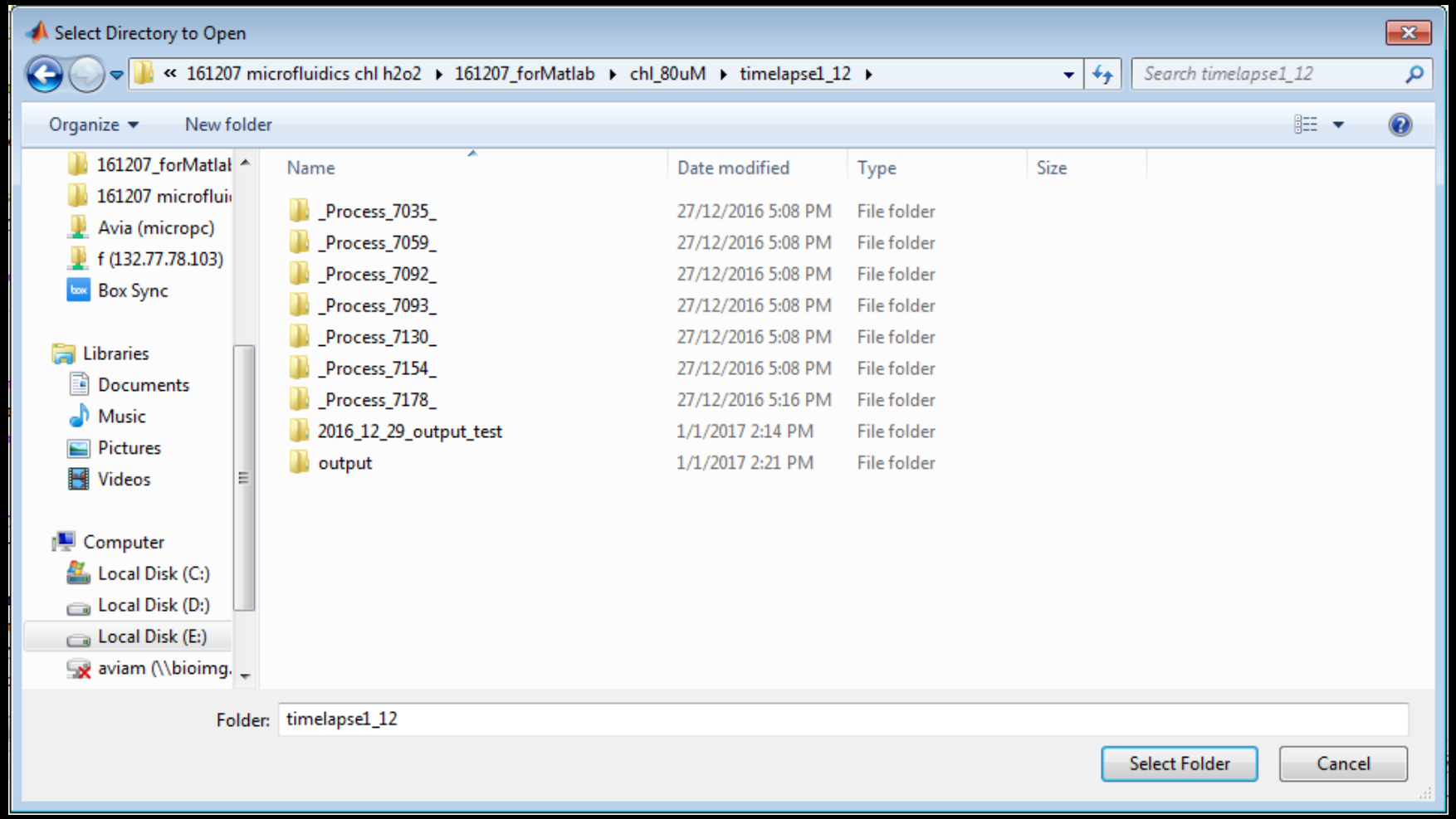
To do so it uses bio-formats:

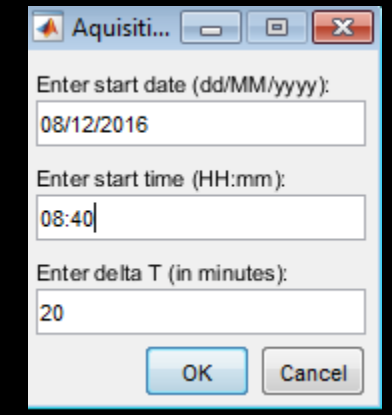
https://www.openmicroscopy.org/site/support/bio-formats5.1/users/matlab/

*Before running the script on a different computer verify that the bio-formats are installed – check out the link above.

Melissa Linkert, Curtis T. Rueden, Chris Allan, Jean-Marie Burel, Will Moore, Andrew Patterson, Brian Loranger, Josh Moore, Carlos Neves, Donald MacDonald, Aleksandra Tarkowska, Caitlin Sticco, Emma Hill, Mike Rossner, Kevin W. Eliceiri, and Jason R. Swedlow (2010) Metadata matters: access to image data in the real world. The Journal of Cell Biology, Vol. 189no. 5777-782 doi: 10.1083/jcb.201004104

2. Open and concatenate the stacks + get time-points data






```
This is the aquisition date of stack 5:
2016-12-08T07:21:27
```

3. Pre-processing: image stabilization (ImageStabilizer)

To correct XY-drifting over time, the script uses the FIJI plugin "image stabilizer":

http://imagej.net/Image_Stabilizer

The stabilization is performed on the **Chl** Channel and then applied to the other channels.

What does it do?

This plugin stabilizes jittery image stacks using the Lucas-Kanade algorithm. It supports both grayscale and color images.

How does it work?

It uses the currently shown slice in an image stack as the initial reference, or "template"; It estimates the geometrical transformation needed to best align each of the other slices with the "template". The estimation and alignment are performed using the Lucas-Kanade algorithm; Once a slice is aligned, the "template" will be updated on the fly using the formula: new_template = a * old_template + (1 - a) * newly_aligned_slice, where "a" is the "template update coefficient" that can be adjusted when the plugin is run.

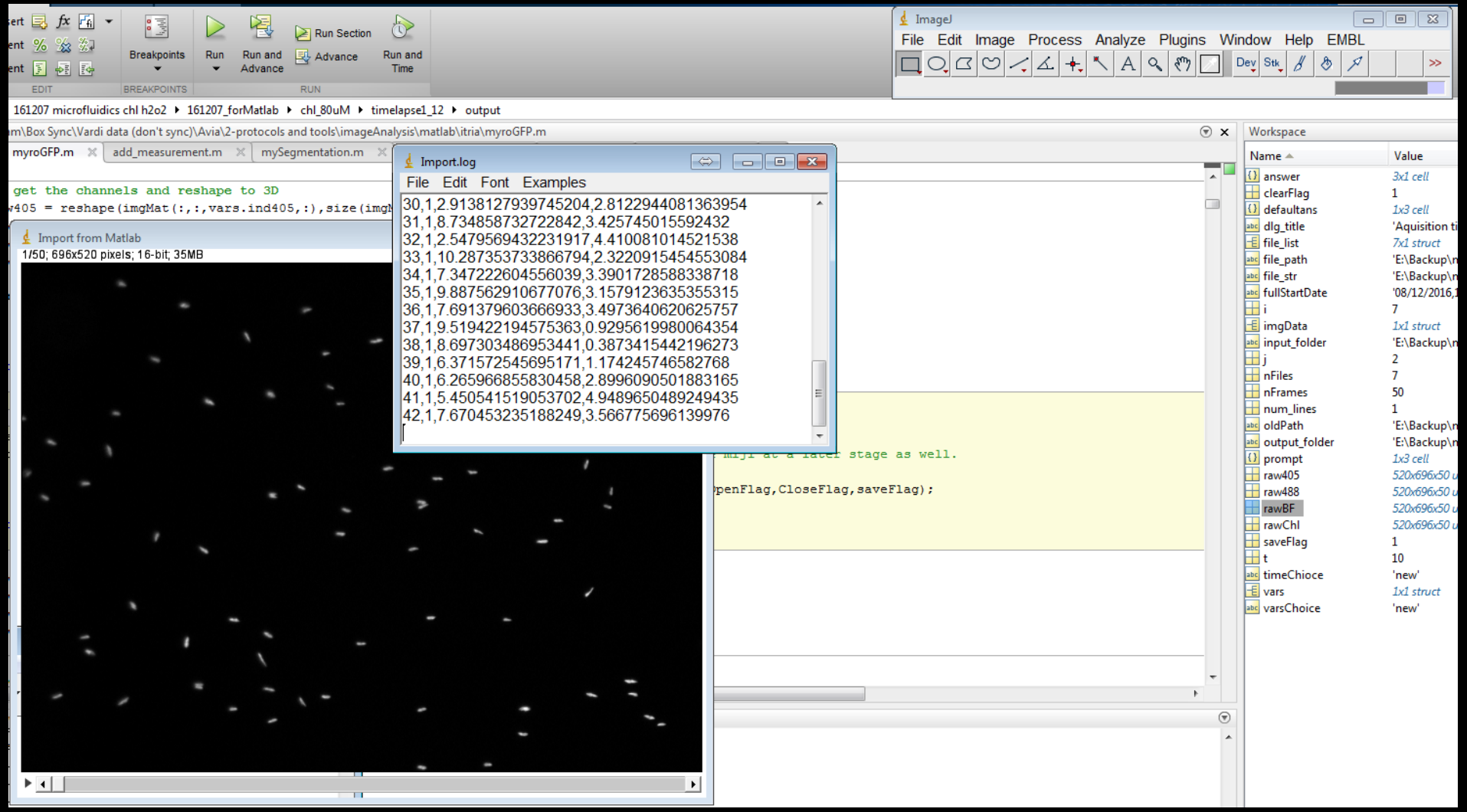
To use this plugin, the Itria script opens FIJI (or ImageJ) from matlab using MIJI:
http://imagej.net/Miji
http://bigwww.epfl.ch/sage/soft/mij/

To run the script on your own computer you have to verify that MIJI is installed correctly, and so do FIJI and the image stabilizer plugin. Follow the instructions in the links above.

K. Li, "The image stabilizer plugin for ImageJ," http://www.cs.cmu.edu/~kangli/code/Image_Stabilizer.html, February, 2008.
Daniel Sage, Dimiter Prodanov, Jean-Yves Tinevez and Johannes Schindelin, MIJ: Making Interoperability Between ImageJ and Matlab Possible ImageJ User & Developer Conference, 24-26 October 2012, Luxembourg.

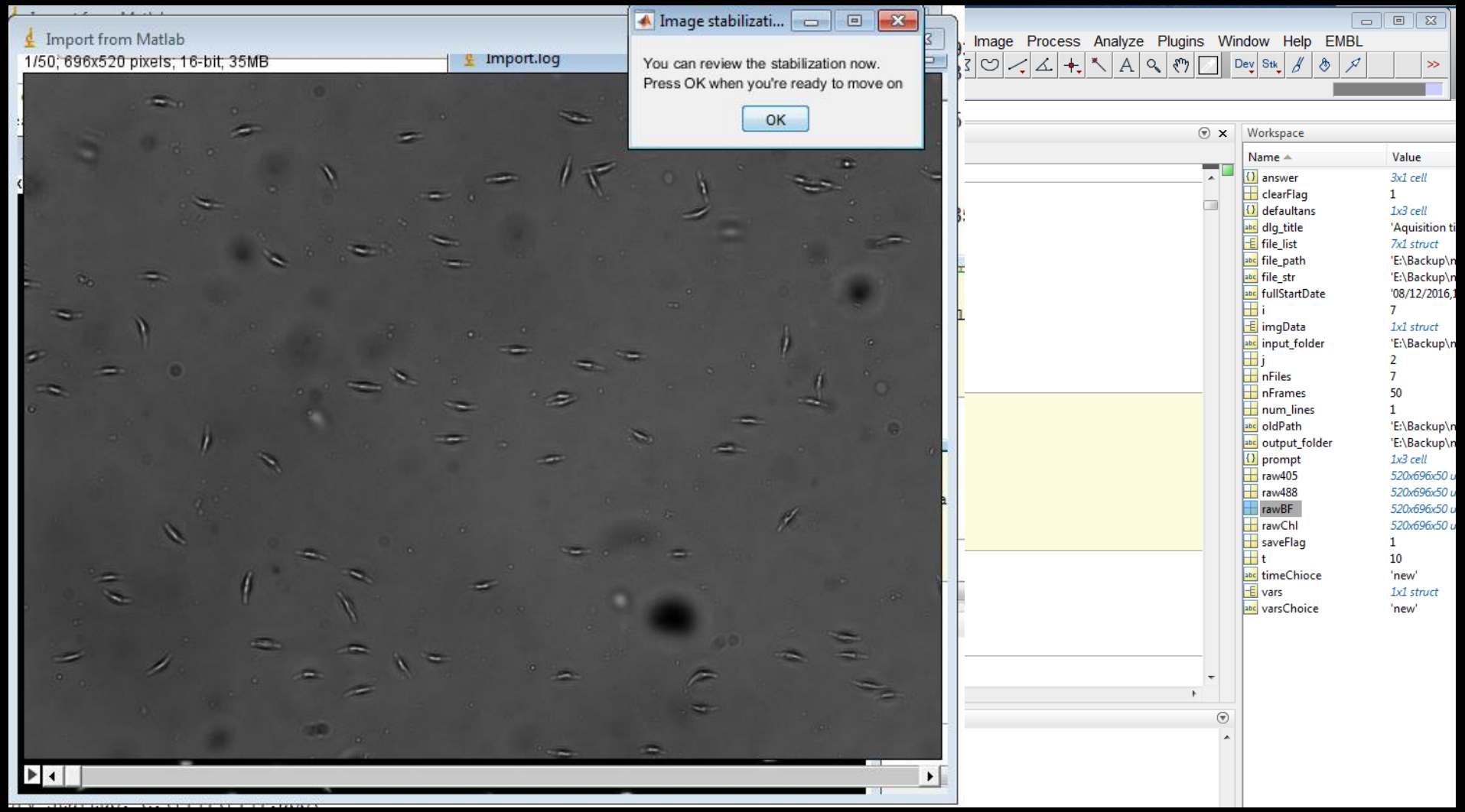
*all examples: chl 80uM H2O2, field 12 in exp 161207

3. Pre-processing: image stabilization (ImageStabilizer)



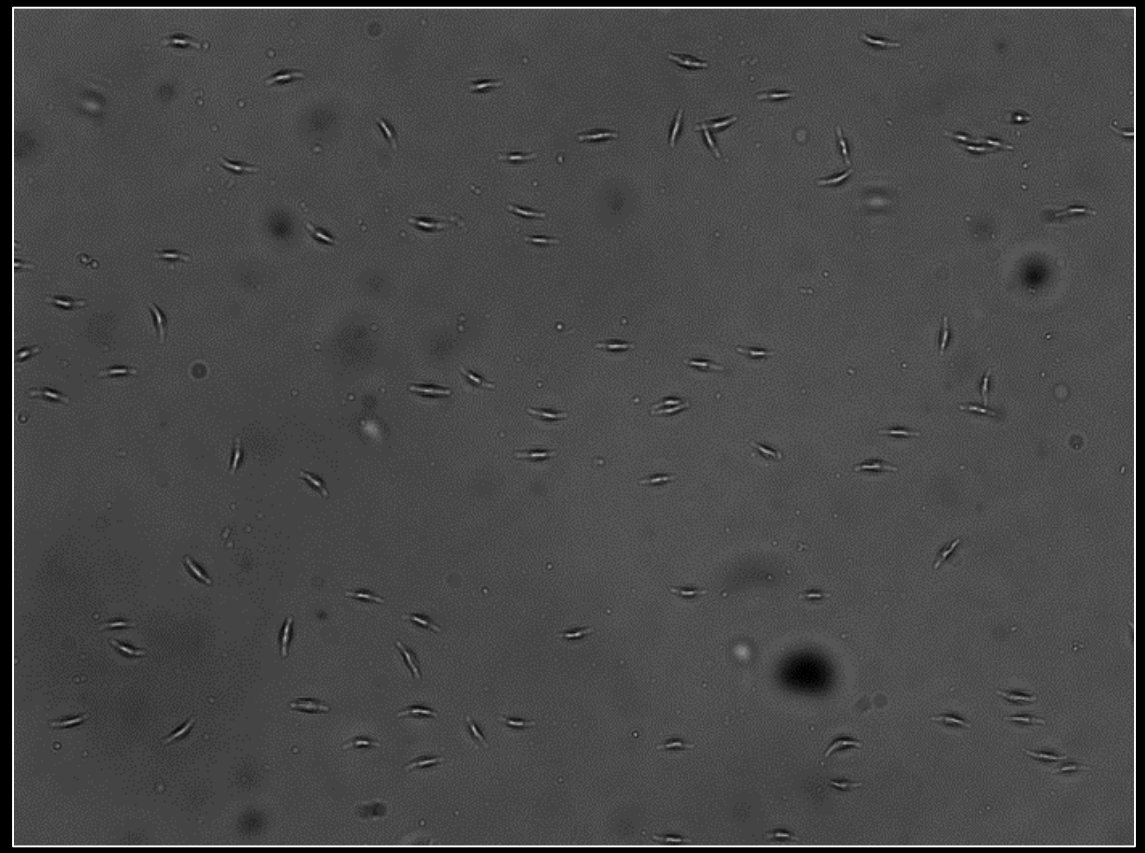
*all examples: chl 80uM H2O2, field 12 in exp 161207

3. Pre-processing: image stabilization (ImageStabilizer)

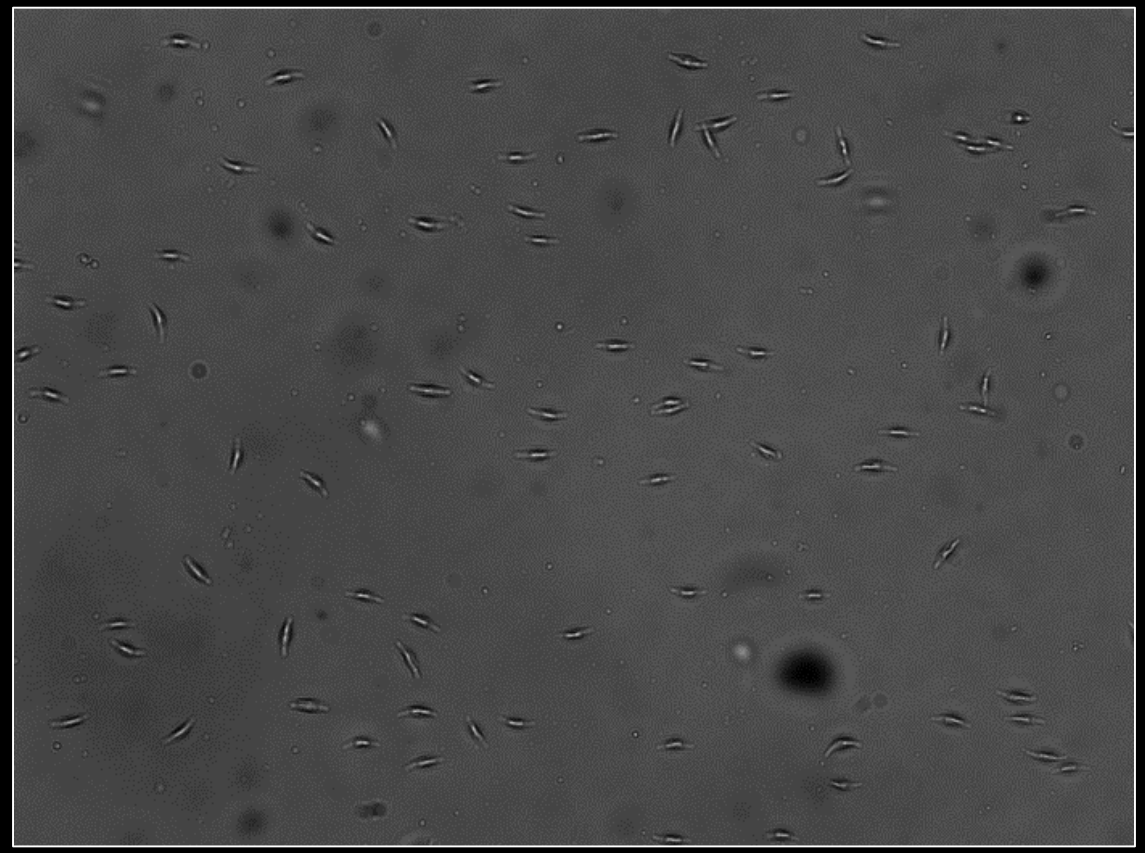


3. Pre-processing: image stabilization

Before

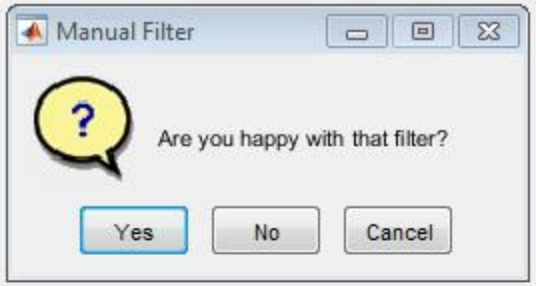


After

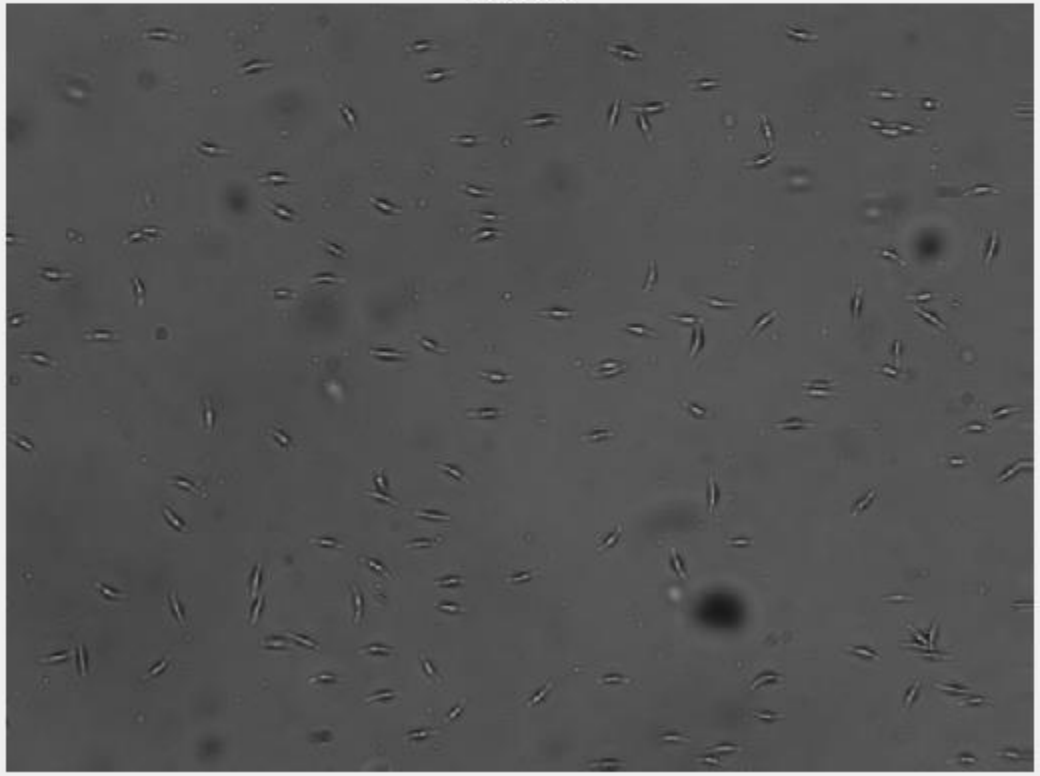


BF band-pass filtering

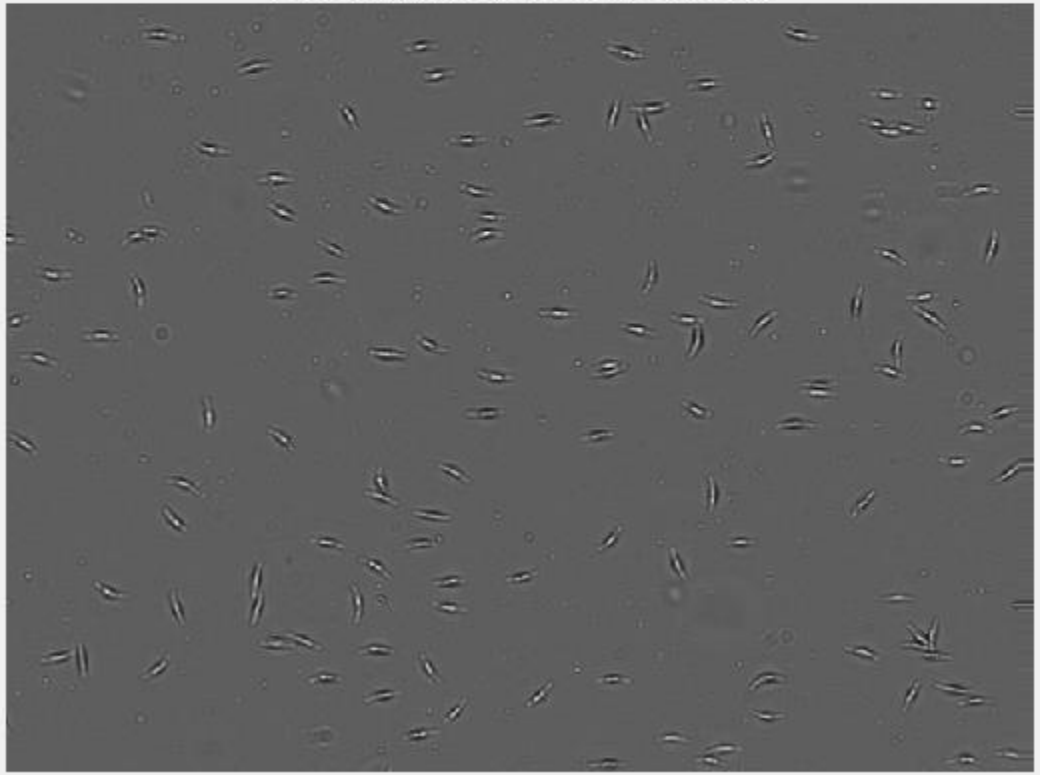
the band-pass filter code is based on a code kindly provided by Vicente I. Fernandez and Roman Stocker with some modifications.



Original

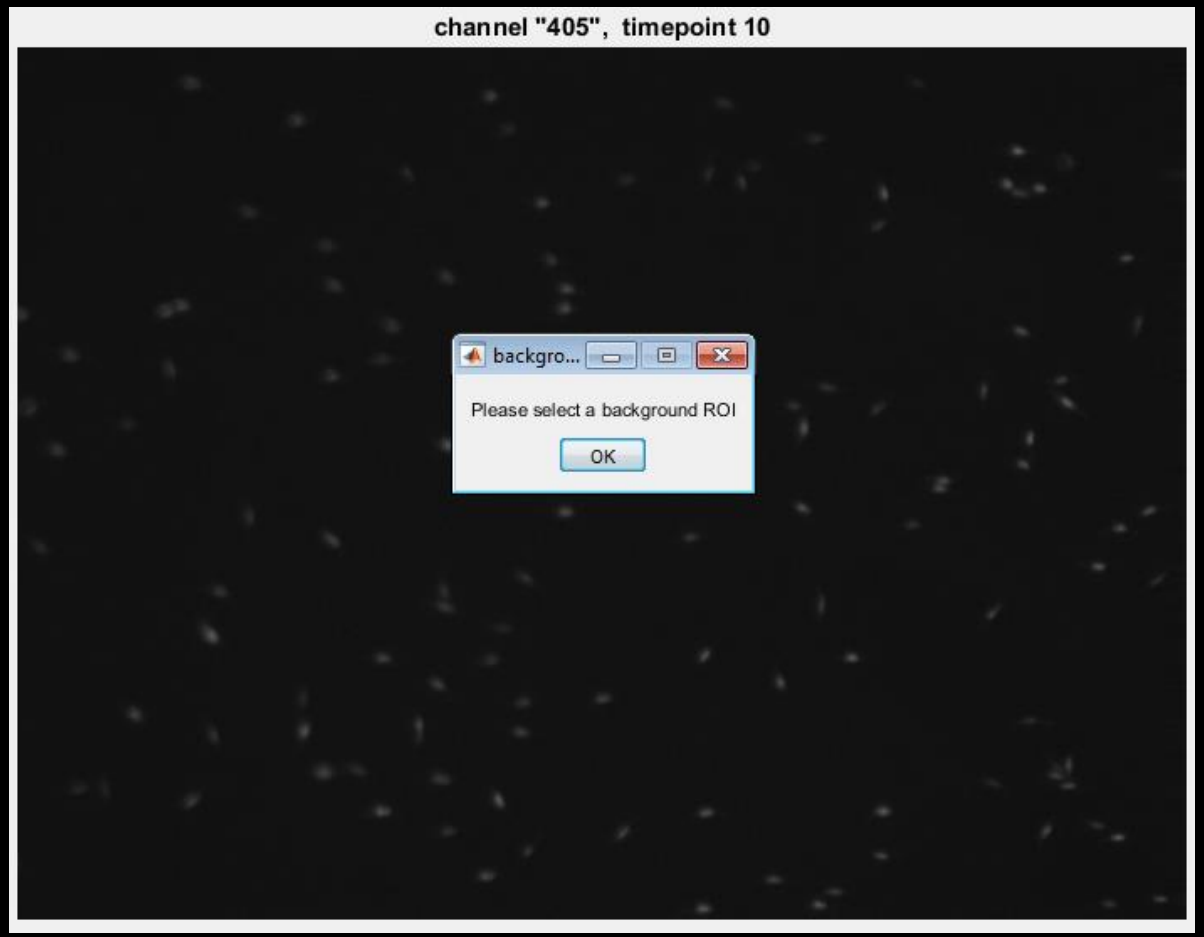


low: 0.2, high: 2, size: 41, Gaussian



Background subtraction
channels: 405, 488, chl

The user chooses a background ROI. The mean value is measured and subtracted from the entire stack. If no area is chosen after several trials than the min value of the stack is subtracted instead.



Problems:

- doesn't account for uneven illumination.
- Assumes the background is the same over time (true for most cases)
- User-dependent (human errors)

Benefits:

- user-defined (no artificial errors).
- Easy to use, rather fast.
- Doesn't manipulate the intensity values

Another option (work in progress):

- Acquire background image for each channel and subtract that.

Background subtraction
channels: 405, 488, chl

The user chooses a background ROI. The mean value is measured and subtracted from the entire stack. If no area is chosen after several trials than the min value of the stack is subtracted instead.

Problems:

- doesn't account for uneven illumination.
- Assumes the background is the same over time (true for most cases)
- User-dependent (human errors)

Benefits:

- user-defined (no artificial errors).
- Easy to use, rather fast.
- Doesn't manipulate the intensity values

Another option (work in progress):

- Acquire background image for each channel and subtract that.

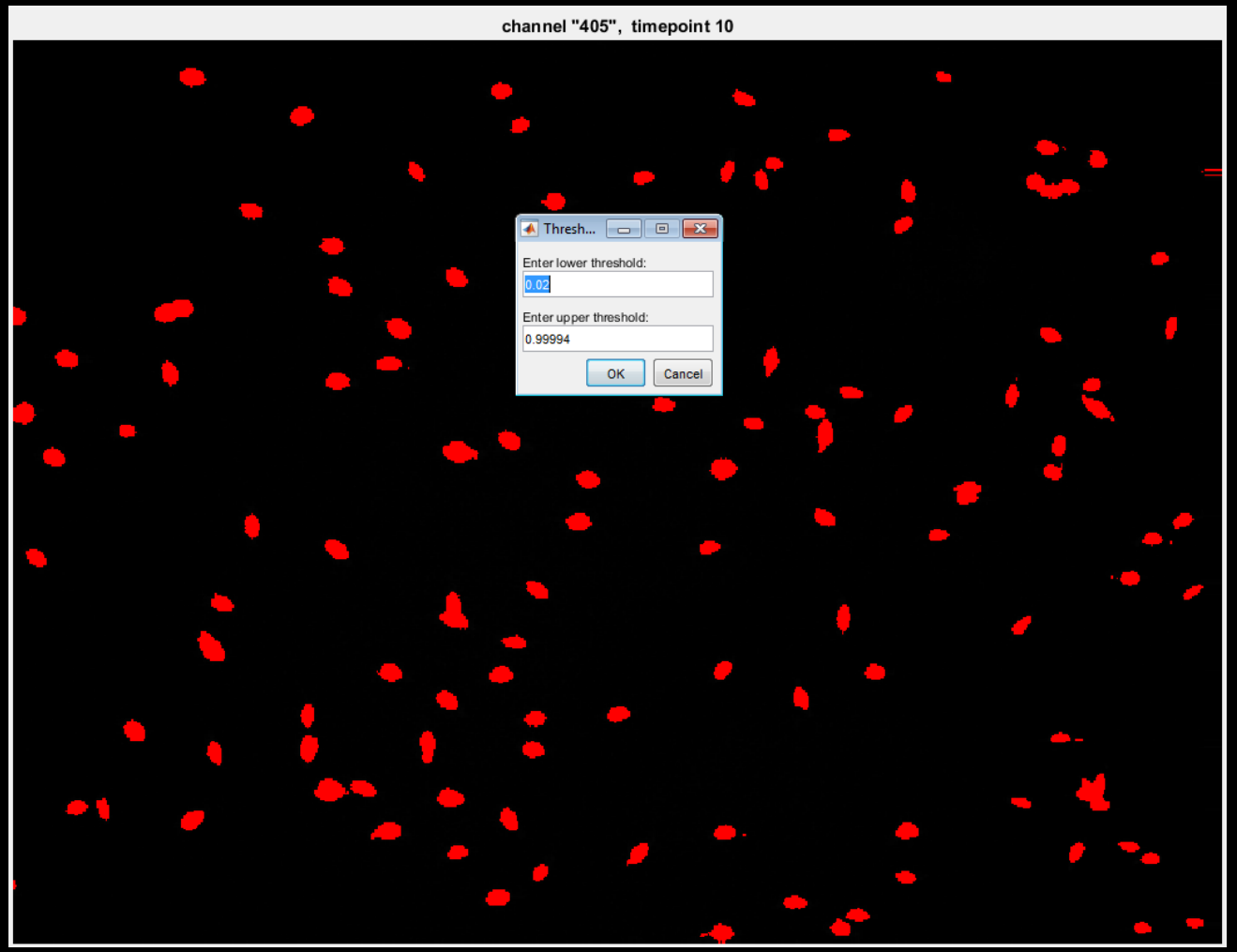
Threshold
channels: 405, 488, chl

For each fluorescent channel the pre-defined threshold is viewed, and the user needs to confirm (in manual mode).

By default, the max threshold excludes saturated pixels.

The user can modify the threshold and view the result before processing the entire stack.

At the end there's an option to view a movie of the mask overlaid on the BF stack.



Red – pixels within the threshold.

Threshold
channels: 405, 488, chl

For each fluorescent channel the pre-defined threshold is viewed, and the user needs to confirm (in manual mode).

By default, the max threshold excludes saturated pixels.

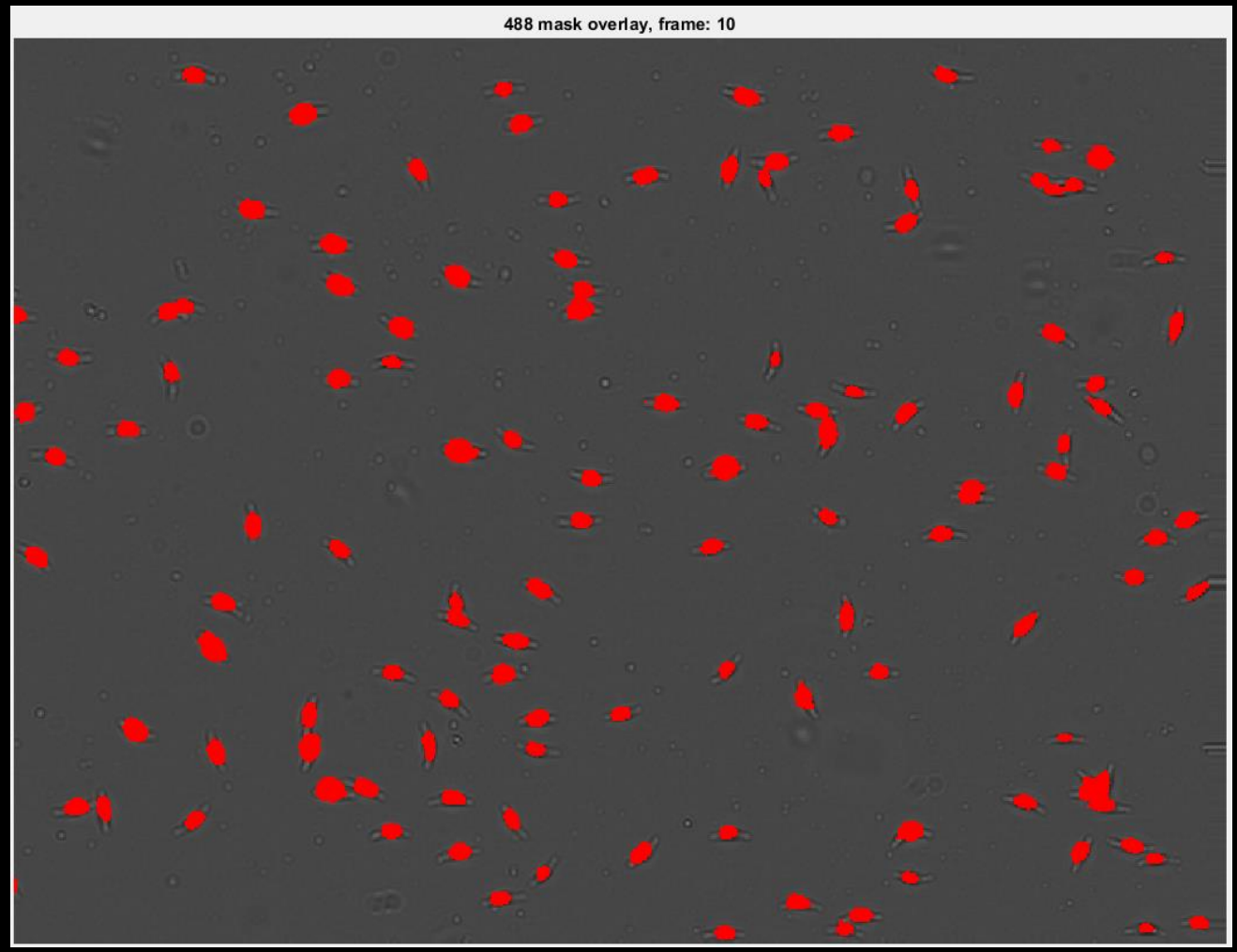
The user can modify the threshold and view the result before processing the entire stack.

At the end there's an option to view a movie of the mask overlaid on the BF stack.

Masks for the tracking:

- roGFP strains: 405_cells, with wider threshold than the roGFP405 threshold.
- Wt: Chl

Red – pixels within the threshold.



View initial results: 405-488 difference false color video

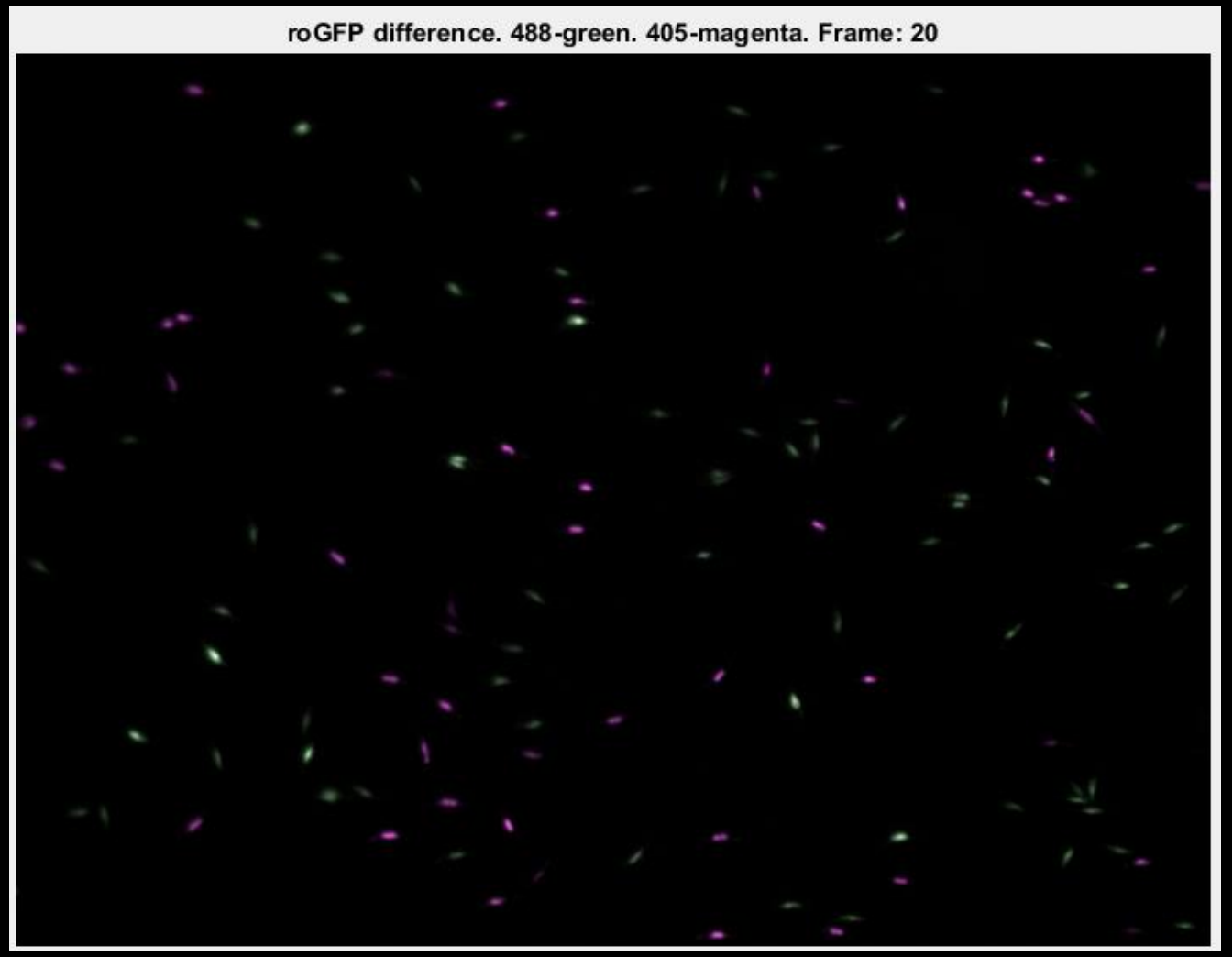
After background subtraction, the **difference** between the 405 and 488 channels is shown in false-color.

It is not the roGFP ratio or any calculation of sort, but can give an initial idea on what happened in the experiment and whether everything worked OK.

It can also show whether the channels are aligned correctly.

Green – higher intensity in 488.

Magenta – higher intensity in 405.



roGFP Masks:
Co-localization & roGFP expression

Co-localization: Create a mask of pixels that "passed" both thresholds: 405 and 488.

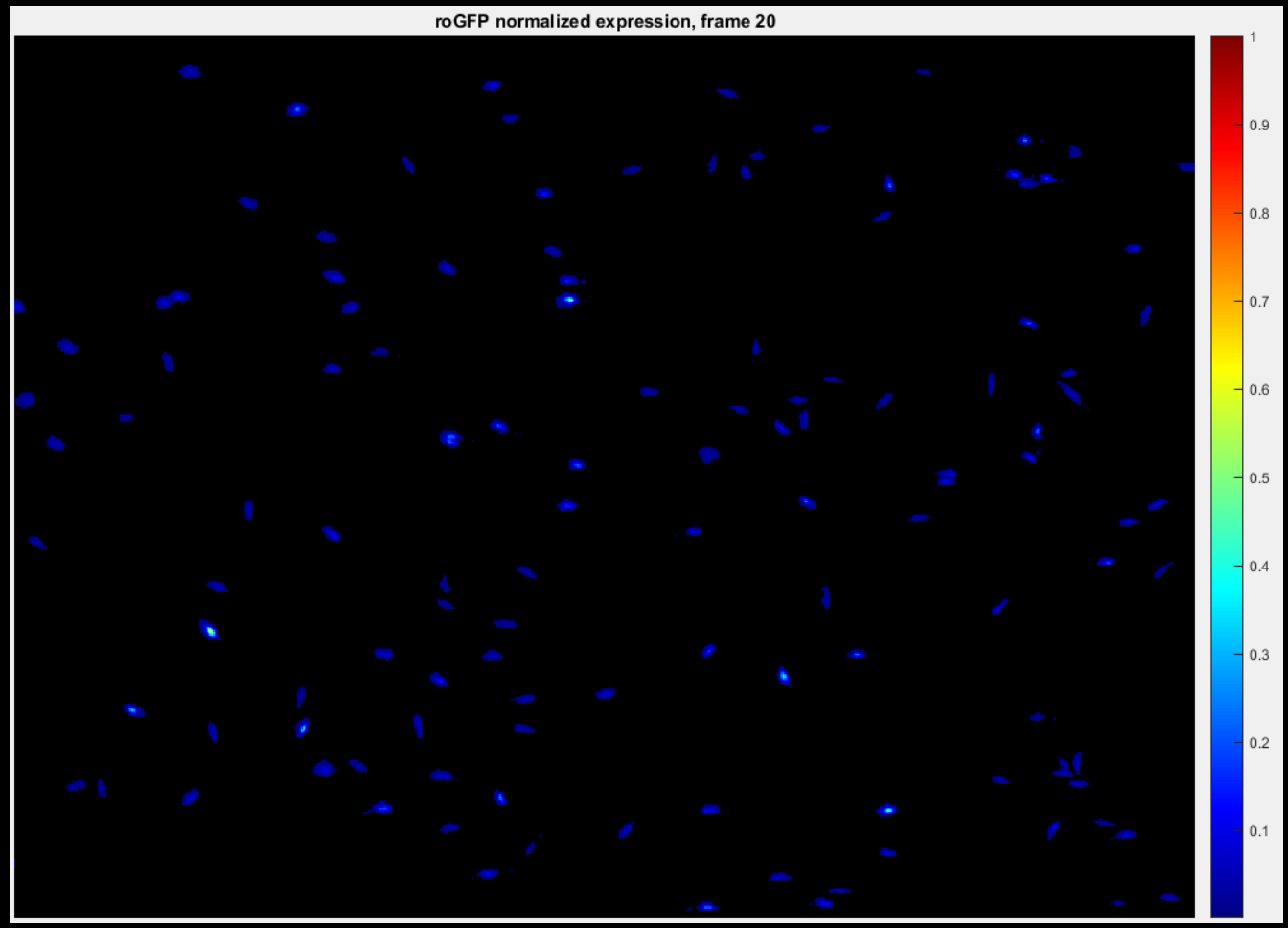
roGFP expression: calculate the expression level pixel-by pixel according to the equation:

roGFP expression = i405 * i488

Then, threshold the expression image so that only high enough expression levels are considered for the analysis.

The roGFP mask in the end is only pixels that passed **3 thresholds**: 405, 488 and expression level.

The discrimination can be more or less strict depending on the threshold boundaries chosen by the user.



roGFP Masks:
Co-localization & roGFP expression

Co-localization: Create a mask of pixels that "passed" both thresholds: 405 and 488.

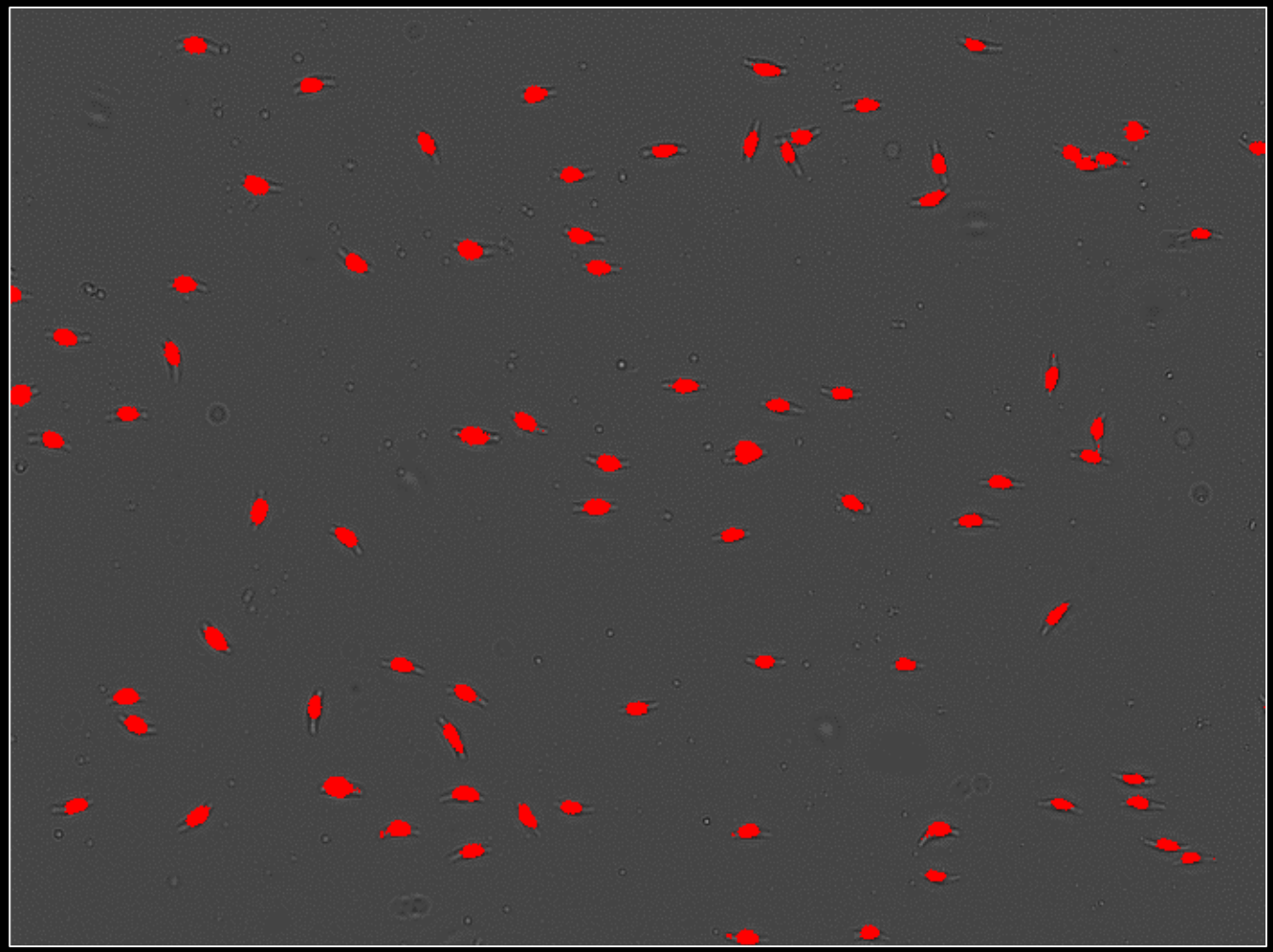
roGFP expression: calculate the expression level pixel-by pixel according to the equation:

roGFP expression = 405i * 488i

Then, threshold the expression image so that only high enough expression levels are considered for the analysis.

The roGFP mask in the end is only pixels that passed **3 thresholds**: 405, 488 and expression level.

The discrimination can be more or less strict depending on the threshold boundaries chosen by the user.



roGFP ratio 405/488

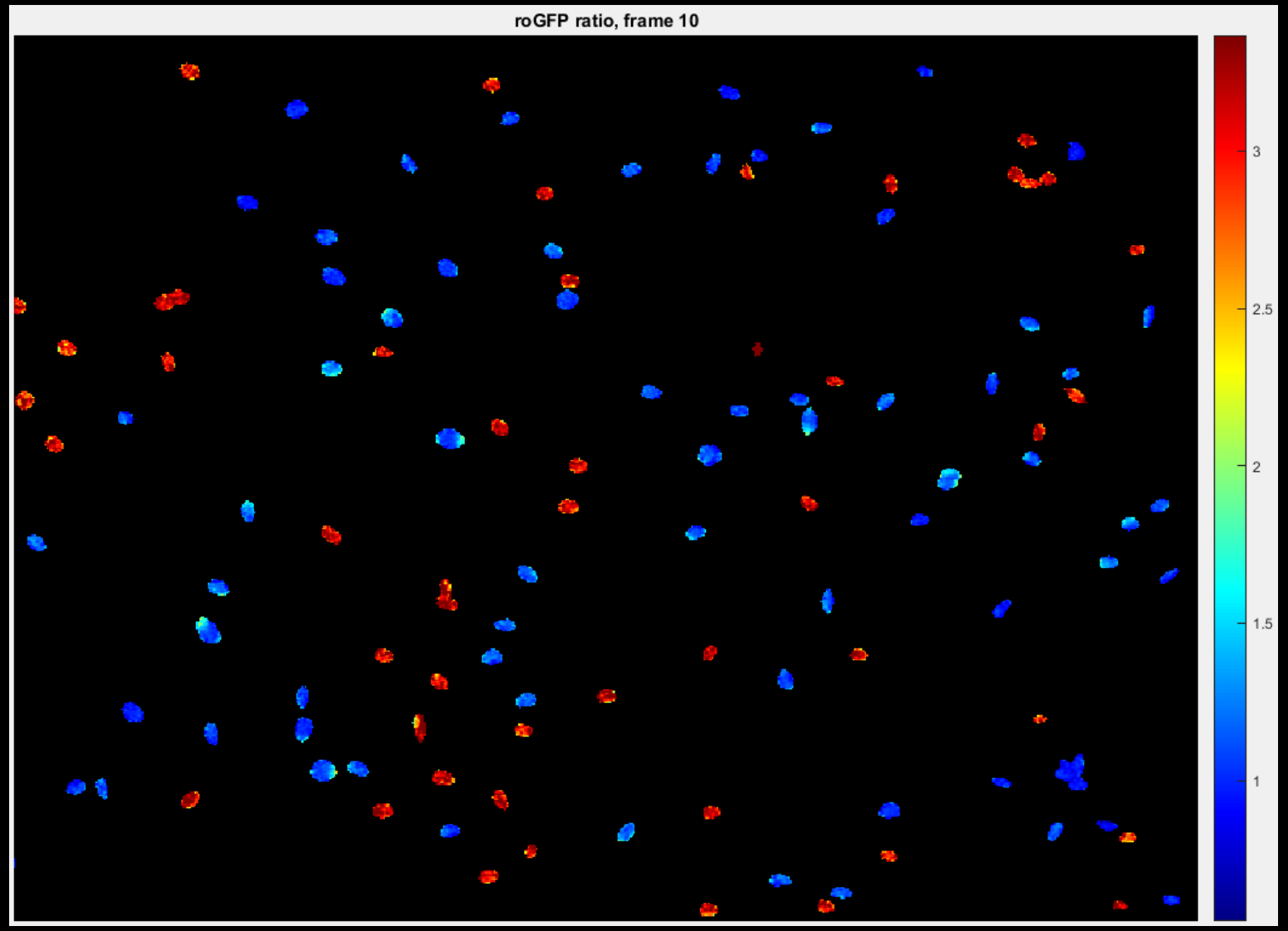
The roGFP ratio is calculated pixel-by-pixel within the roGFP mask:

roGFP ratio = i405/i488

One frame is viewed in false-color before moving on.

Red – oxidized

Blue - reduced

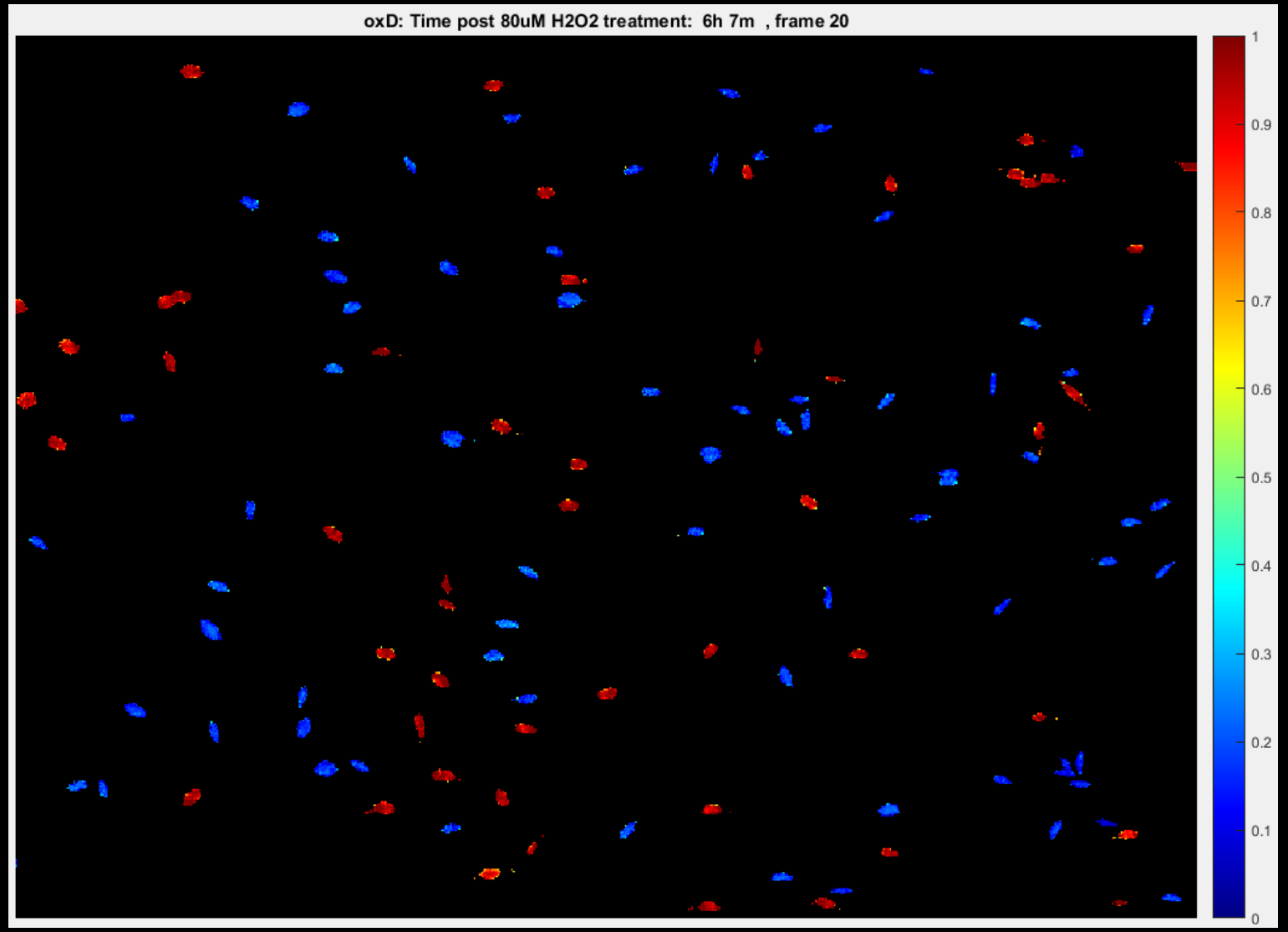


roGFP oxidation (oxD)

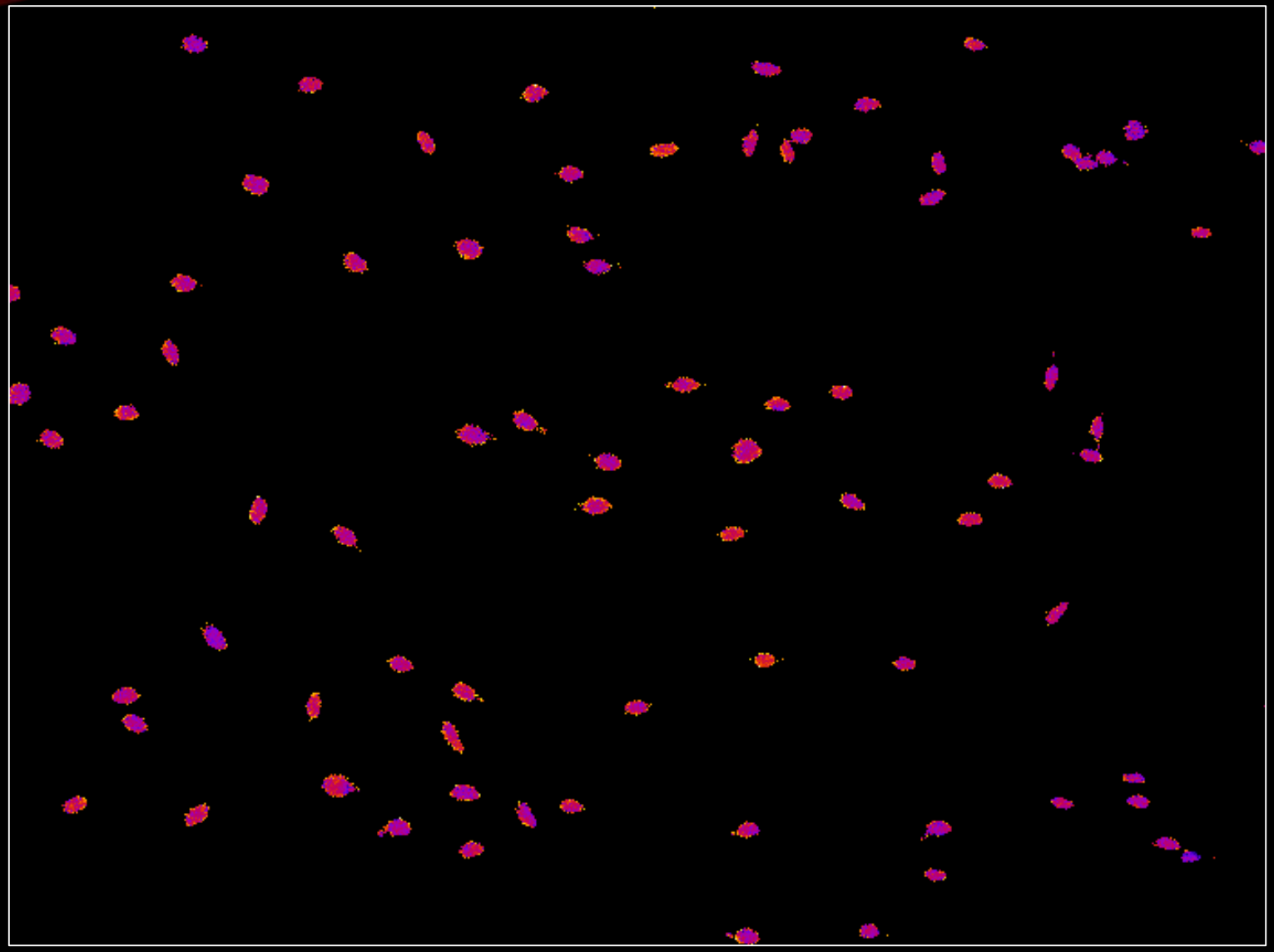
oxD is calculated pixel-by-pixel according to the equation:

$$oxD = \frac{R - Rred}{\frac{i488_{ox}}{i488_{red}} * (Rox - R) + (R - Rred)}$$

Then, a movie of oxD in false-color is shown.

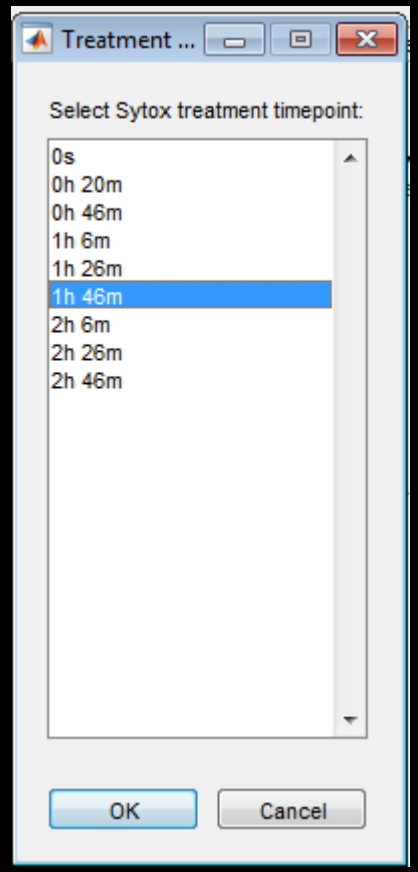


roGFP oxidation (oxD)



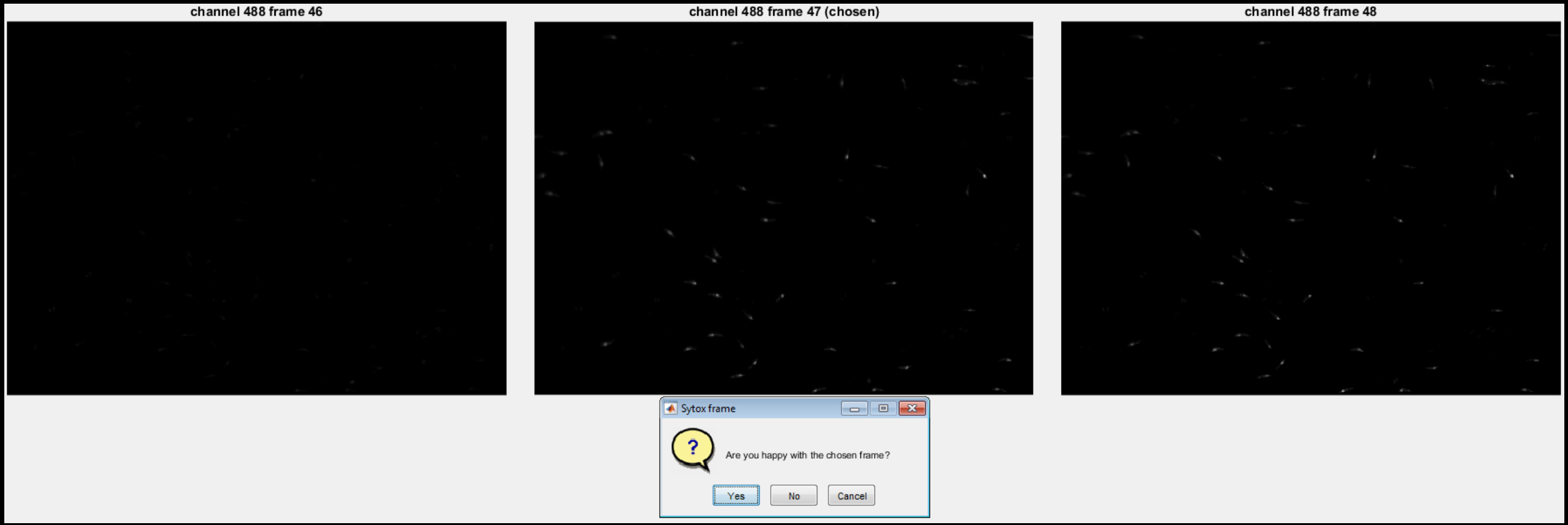
Sytox:
getting the right time-point & threshold

1. Select the time-point for Sytox treatment (time post Sytox treatment, as entered in the "vars" structure).



Sytox:
getting the right time-point & threshold

2. Verify that the right time-point was chosen.
   The 488 channel of the chosen frame and the frame before and after are shown.
   The user can verify or change to a different time-point.

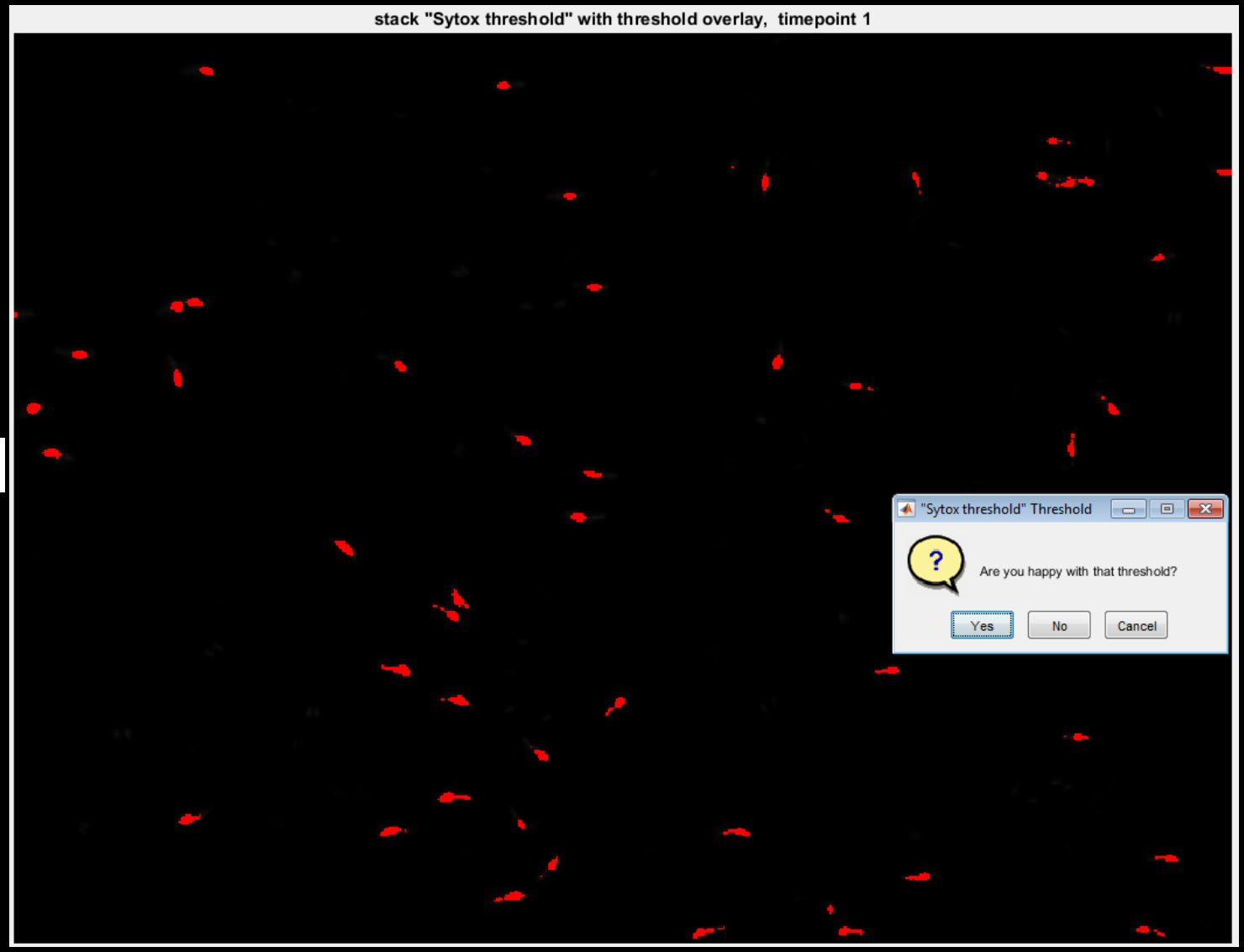


Sytox:
getting the right time-point & threshold

1. **Select the time-point** for Sytox treatment (time post Sytox treatment, as entered in the "vars" structure).
2. **Verify that the right time-point** was chosen. The 488 channel of the chosen frame and the frame before and after are shown. The user can verify or change to a different time-point. After verification, the time-point chosen is printed.

```
Sytox treatment started at 08-Dec-2016 10:21:00, frame: 42
The frame chosen for sytox analysis is: 47,  1h 46m post sytox treatment
```

3. **Threshold the Sytox signal**. The Sytox signal is much stronger than the roGFP488 signal, the two signals can be discriminated. For Sytox imaging I'm using shorter exposure for the 488 channel, meaning that the roGFP signal is much weaker than before. The highest Sytox signal is in the nucleus, but it also spreads to other parts of the cell.



Sytox:
dead or alive?

In order to determine whether a cell is stained with Sytox or not (dead or alive) the script searches for overlap between the Sytox mask and the **dilated** cell mask. The cell mask, based on roGFP405 intensity in roGFP strains (chl for WT), is dilated by 2 pixels to all directions. Pay attention that the 405 channel also represents additional signals: auto-fluorescence and leakage of the Sytox signal. Therefore, the Sytox positive cells are already expected to have bigger cell mask.

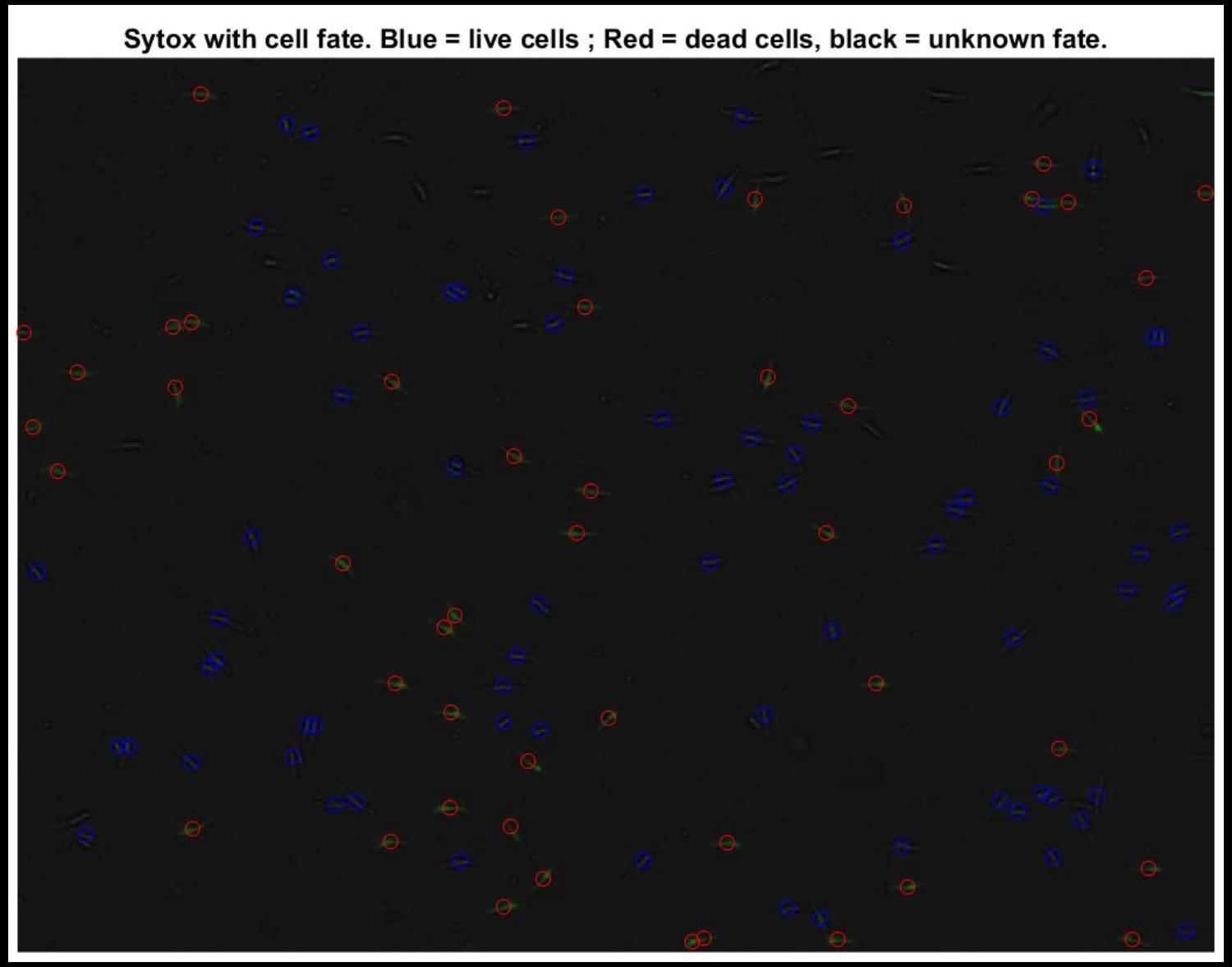
If enough pixels are both in the cell mask and in the Sytox mask, the cell is considered **dead**.
At the moment, the minimum is set to 2 pixels.

If there isn't enough overlap the cell is considered **alive**.

If a cell was not detected in the Sytox frame then the fate is considered **unknown**.

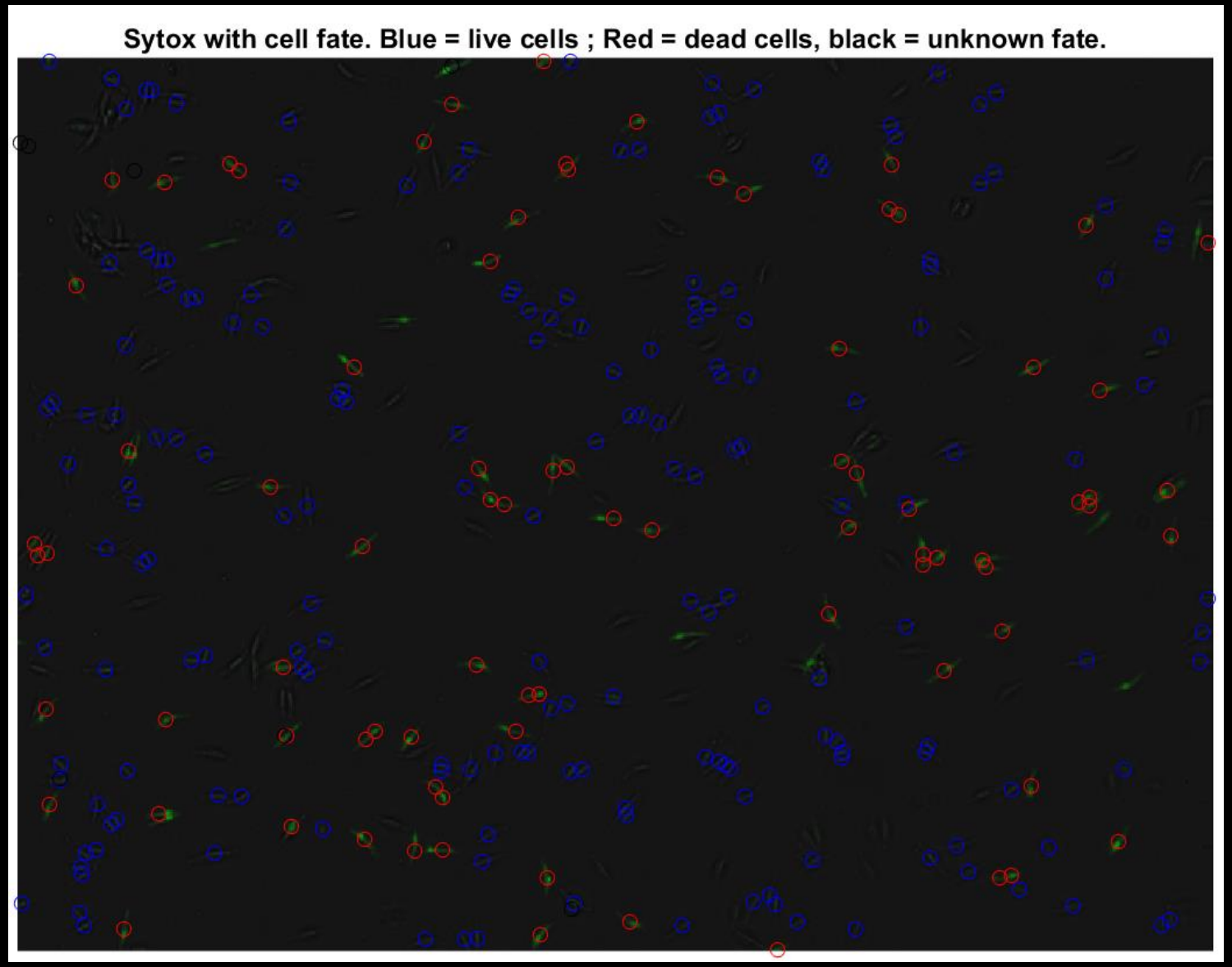
Sytox:
dead or alive?

Field 12



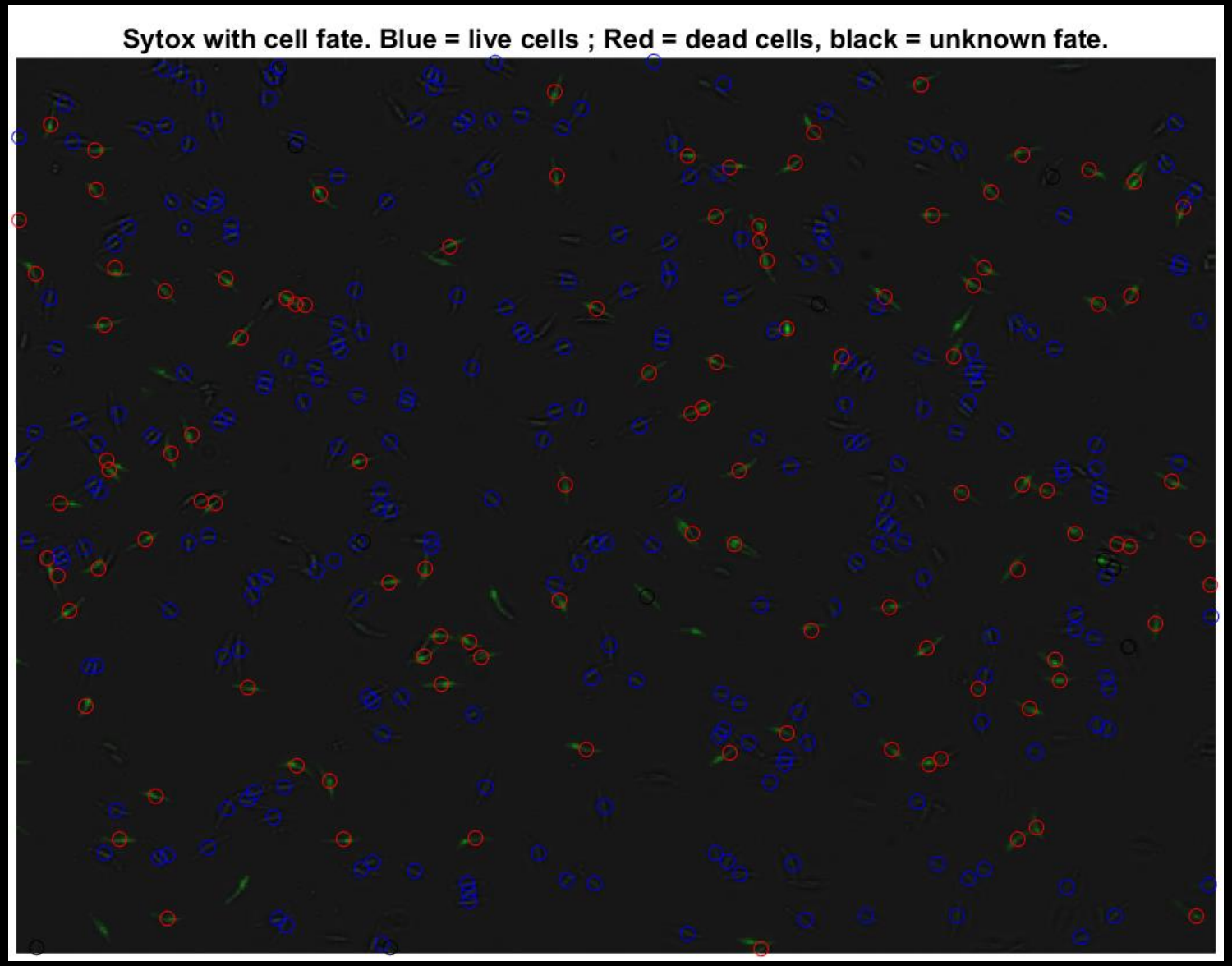
Sytox:
dead or alive?

Field 11



Sytox:
dead or alive?

Field 10



Cell segmentation
separating touching cells

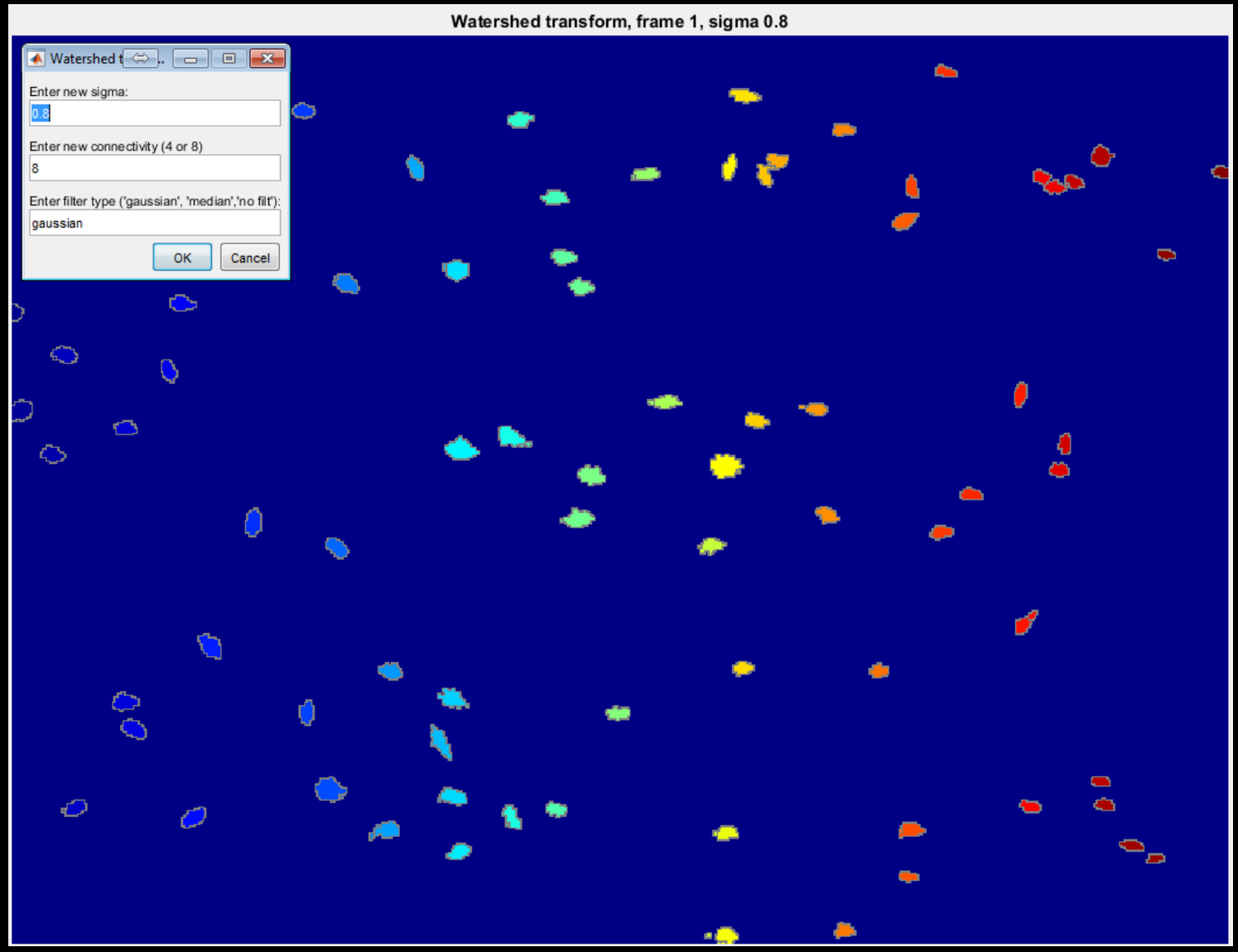
There are several options for cell segmentation, at the moment I'm using the first one:

1) **'mySegmentation'** – Uses a mask and an intensity image. There are 3 modes: distance, intensity or combined. In all modes, the script is trying to separate regions in the image based on the **Watershed** function. Basically, it looks for the borders between the regions.
    1) **Distance** – find local maxima in the intensity stack (for example 405 channel, roGFP expression) and calculate the distances from that.
    2) **Intensity** – use the reciprocal (or multiplicative inverse) fluorescence intensity as a distance matrix to find borders. **This is the option used now.**
    3) **Combined** – normalize the first two options, average them, and used the mean matrix to find the borders. The problems is that the steepness of change in the two values is not the same all over the image, and the average is not taking into consideration different weight for each. In the end, this is not very informative and gives similar results to the "distance" option.
2) **'Fiji'** – uses only a cell mask. It opens FIJI from MATLAB (requires MIJI) and performs the FIJI functions of "erode" and "watershed" (a bit different from the watershed function in MATLAB).
3) **'Mask only'** – The segmentation mask is used as cell mask without cell separation. This option is relevant if the threshold is enough to separate everything, or if you don't mind excluding later any object that is too large (such as touching cells).

mySegmentation example

Stages of segmentation:

1. **Image filtering**: Gaussian, median or no filter. The Gaussian (or median) filter "smoothes" the image, helping to avoid over-segmentation.
2. Calculate the "**distance matrix**" depending on the mode. In the case of Intensity, the distance is the reciprocal value:
   Img_inverted = 1./Img
3. **Watershed** transformation

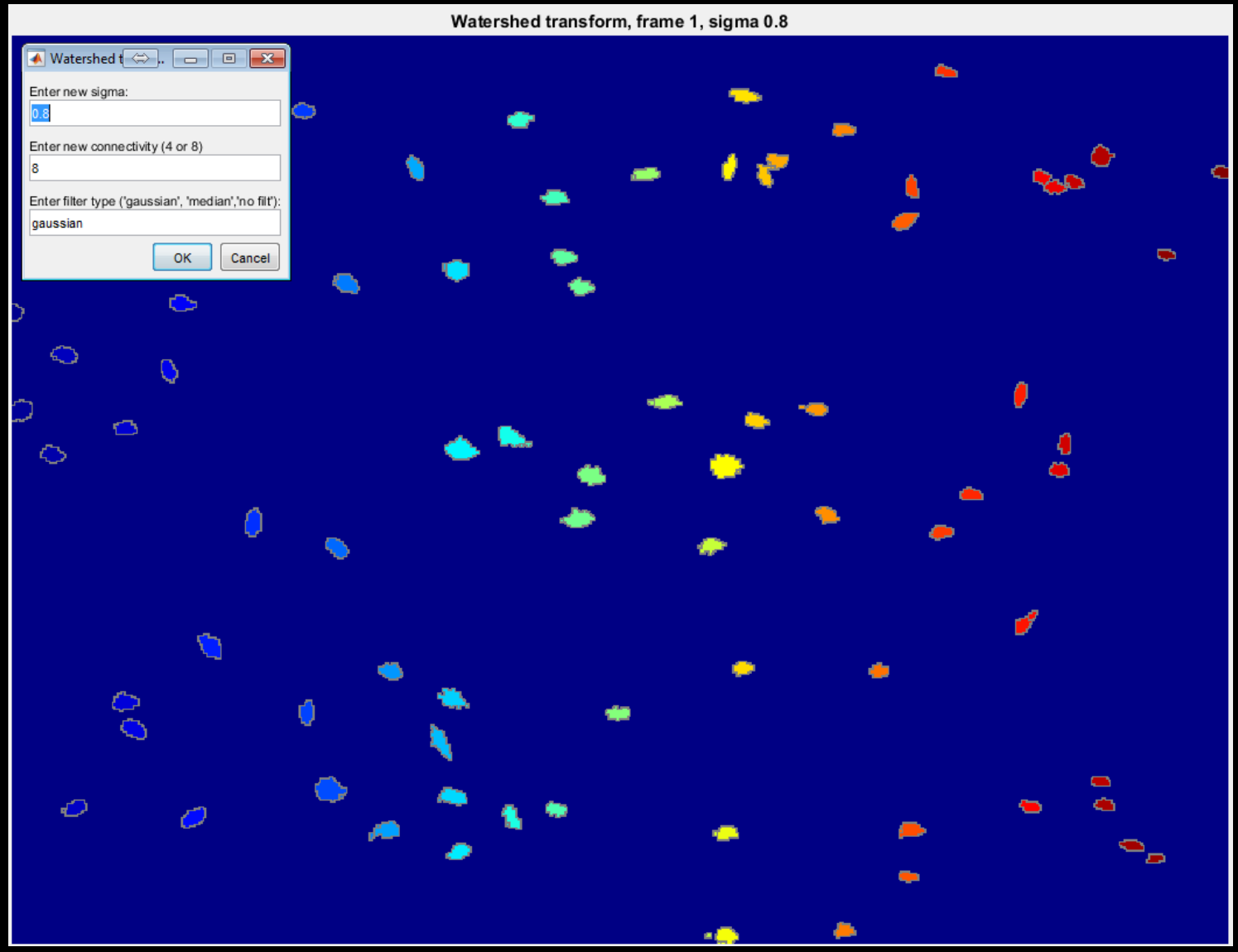


mySegmentation example

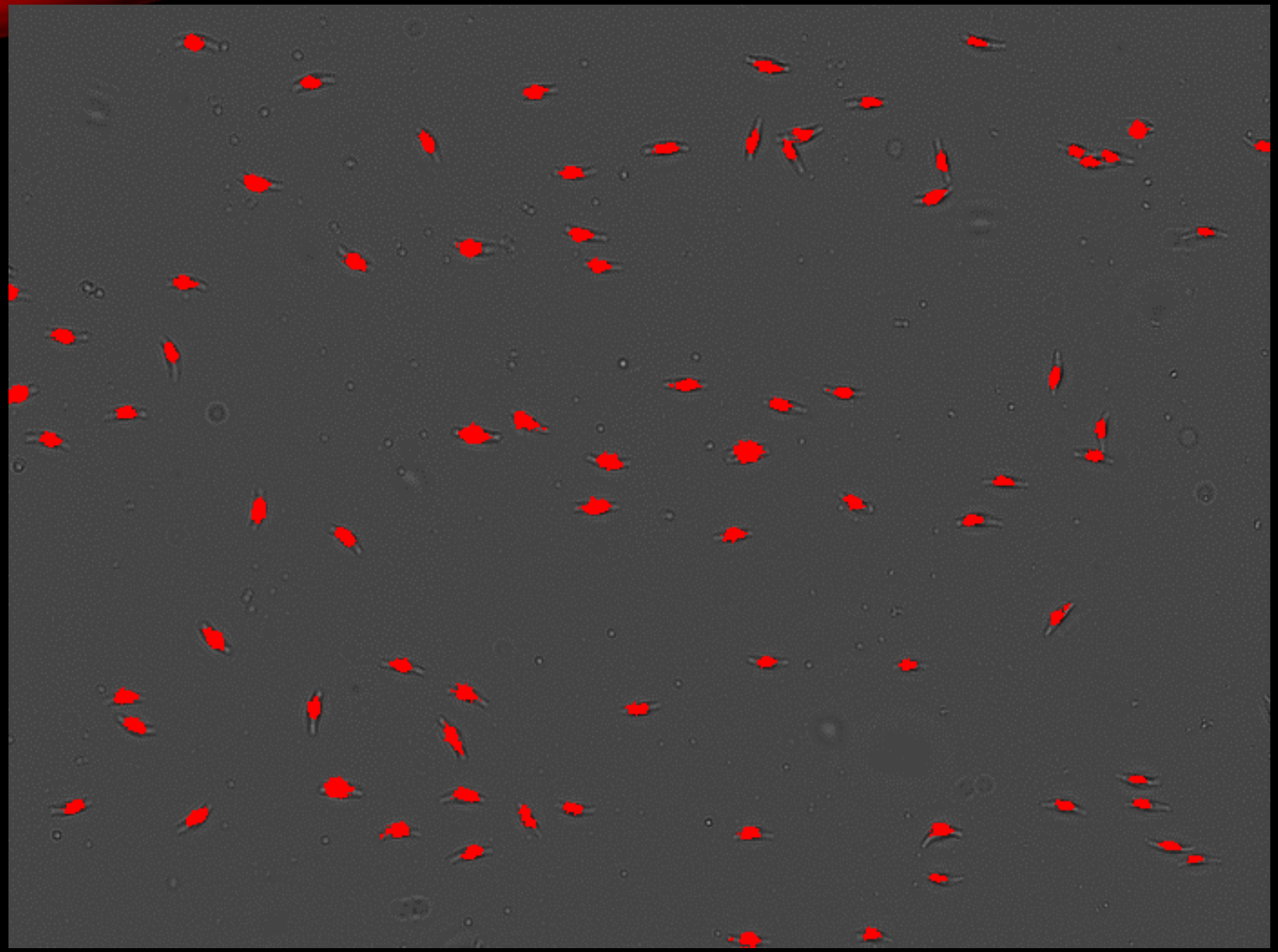
The first image of the stack is segmented and viewed for user verification. Several parameters can be changed:

- **Sigma**: the sigma used for the Gaussian filter (applied on the intensity stack). You can just change it and see what happens, at the moment 0.8 gave the best results…
- **Connectivity**: either 4 or 8. The connectivity used for the watershed function. Basically, it determines the neighbor pixels that are considered in the calculation: whether to include also diagonal neighbors (8) or not (4). You can play with it and see what happens.
- **Filter type**: which type of filter to apply before calculating the distance matrix: Gaussian filter, median filter or no filter.

After confirmation, the whole stack is segmented. The middle frame of the stack is also viewed and parameters can be modified, but it's not recommended to change parameters in the middle of the stack.

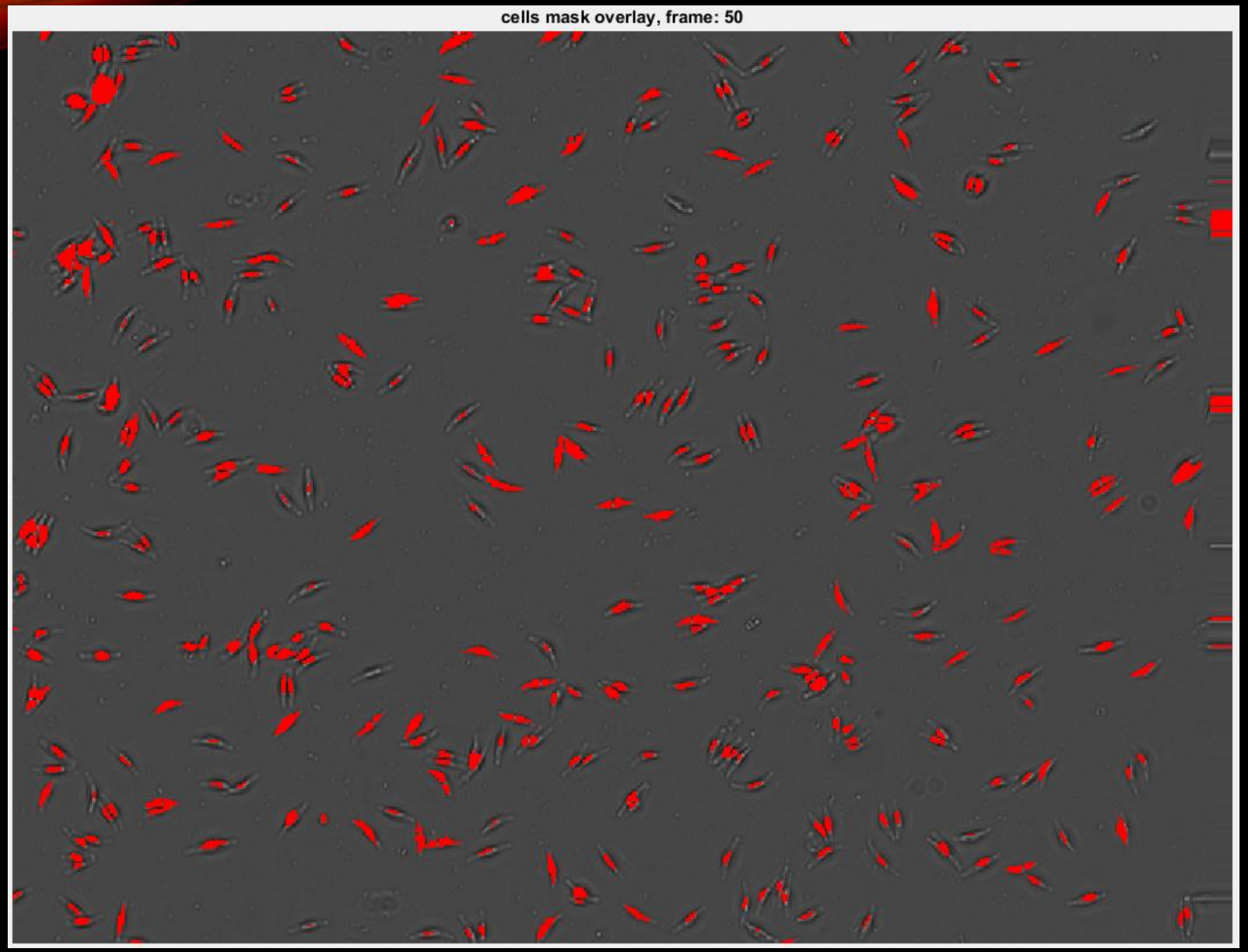


mySegmentation example



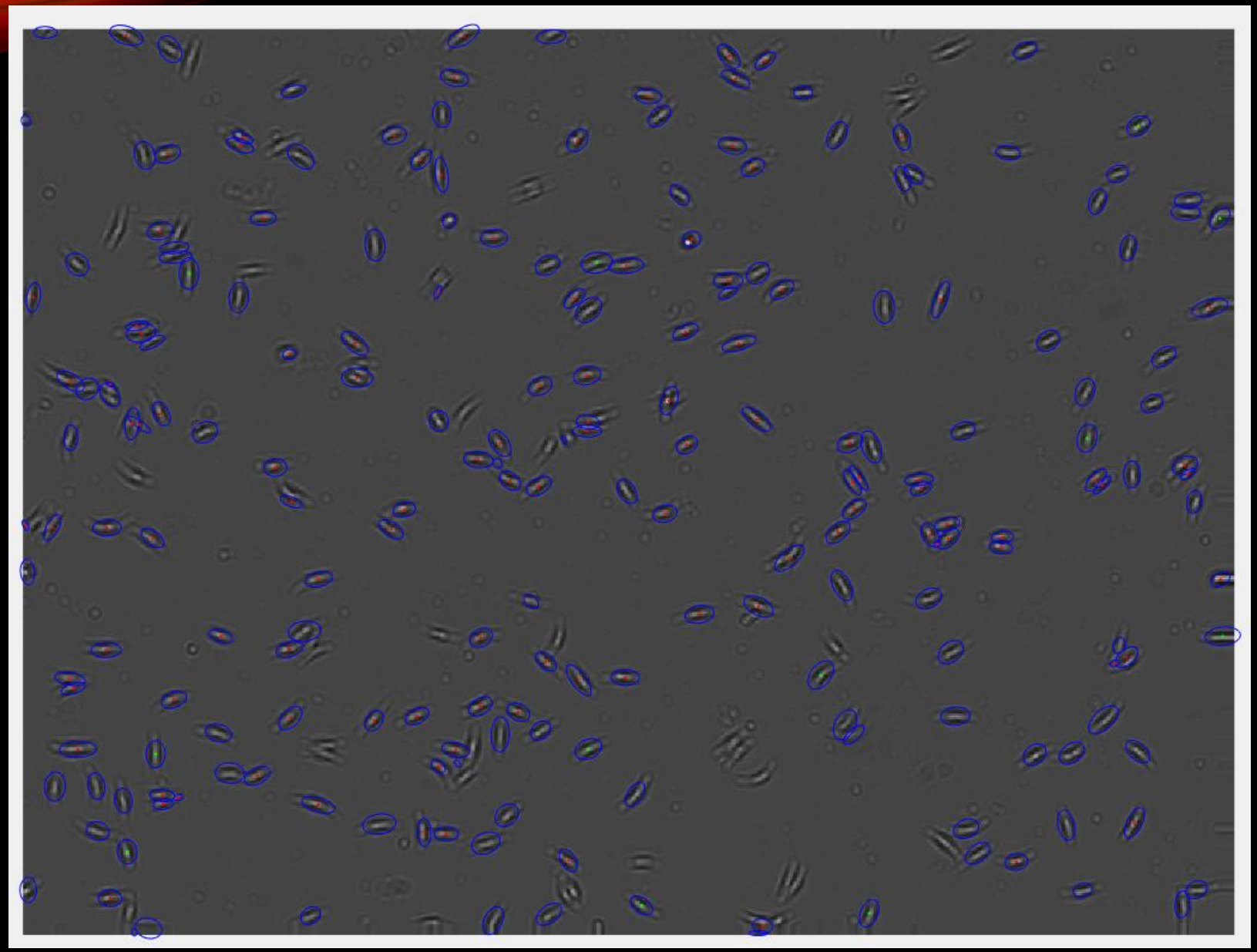
mySegmentation example

Different field:
Chl 80uM H2O2 field 11



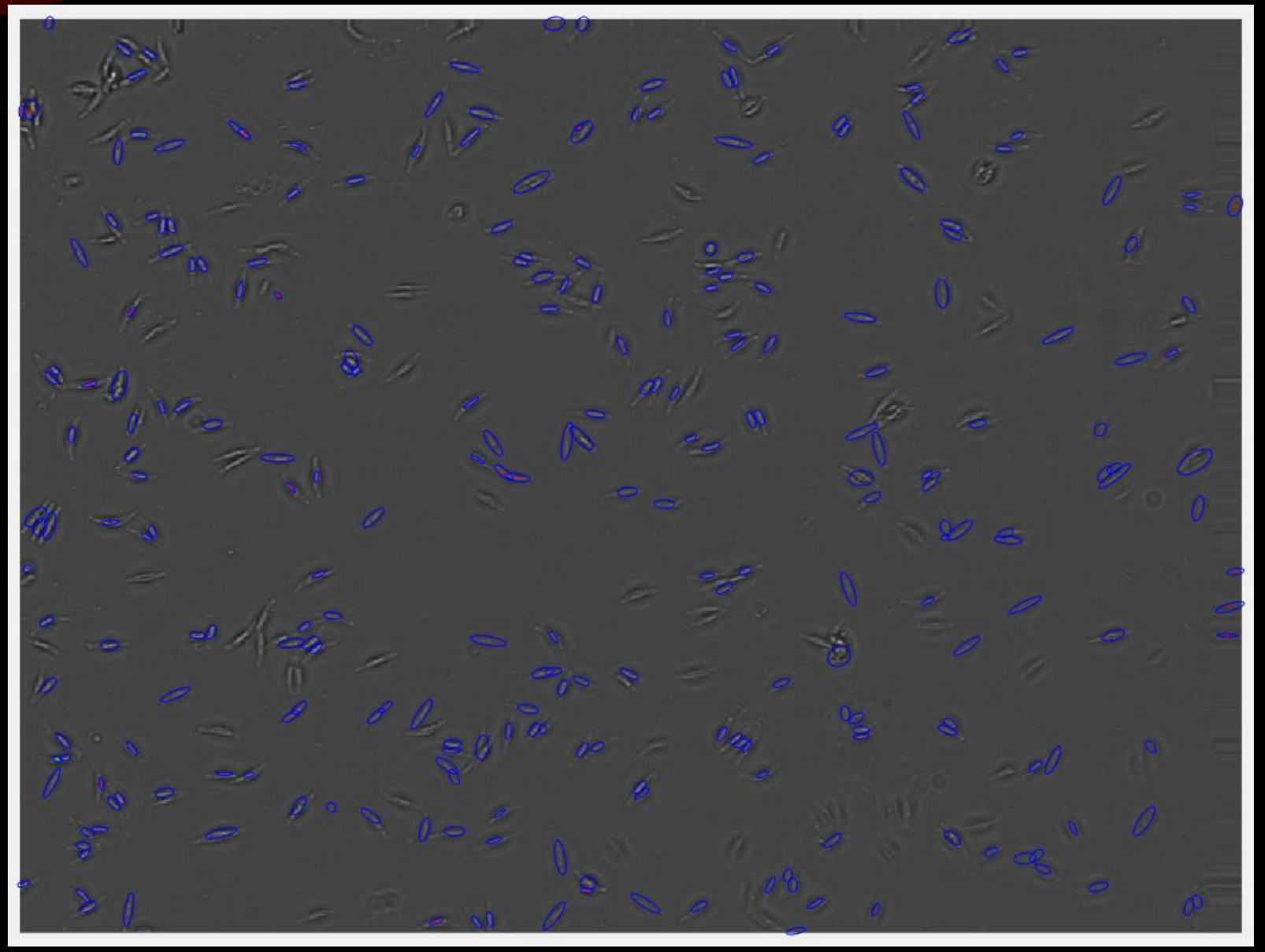
mySegmentation example

Different field:
Chl 80uM H_2O_2 field 11



mySegmentation example

Different field:
Chl 80uM H2O2 field 11

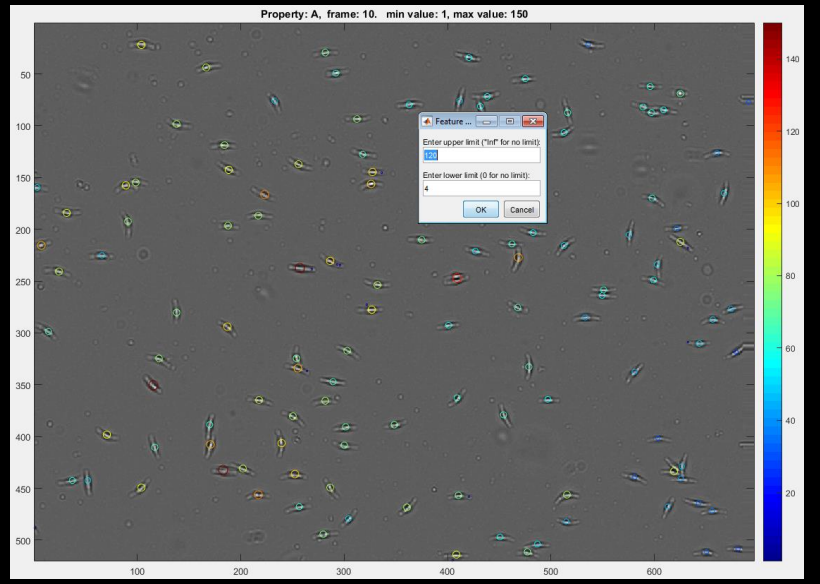


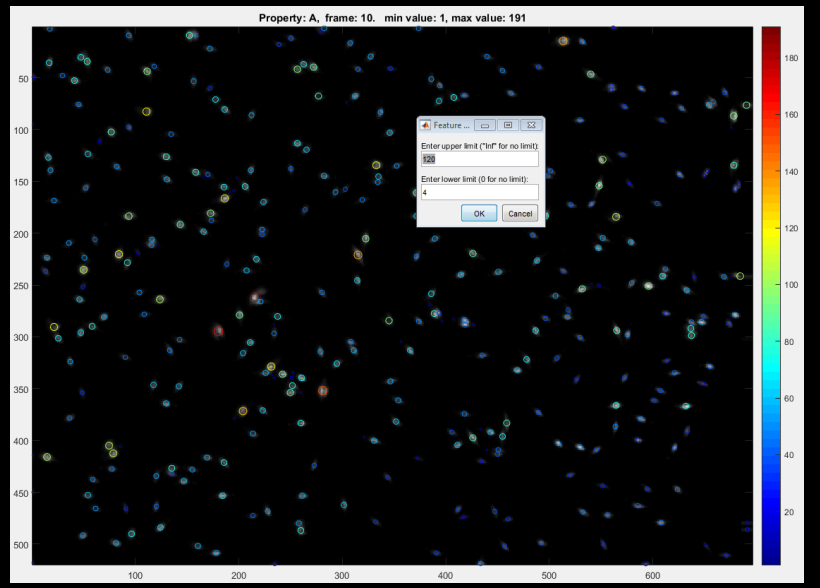
Cell properties and filtering

After segmentation, the cells mask is used to measure various properties of the cells. Then, before tracking, detected objects that are not single cells or that are not in focus can be filtered out using these properties. For example:

- **Area** – filter out very small and very large objects.
- **Eccentricity** – filter cells according to how "round" they are. Values range from 0 (perfect round) to 1 (not round, straight line).
- **Major and minor axis**: use specific cellular features, for example to exclude doublets with a minor axis that is too large.
- **Intensity**: not used at the moment since the intensity and expression are being thresholded before. However, it might be useful for other scenarios.

From this part until the analysis part, the code is based on a code kindly provided by Vicente I. Fernandez and Roman Stocker with some modifications.

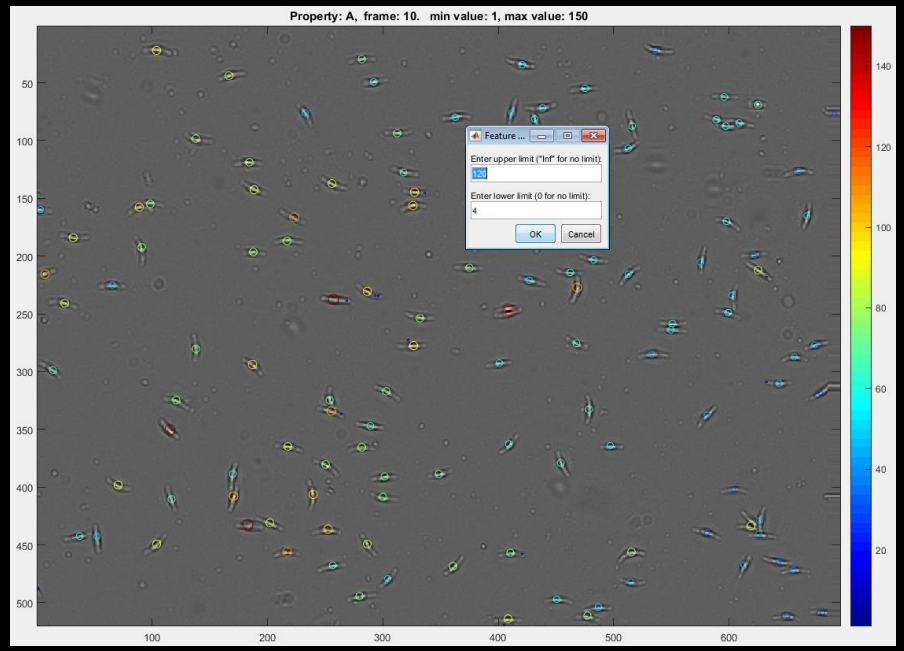


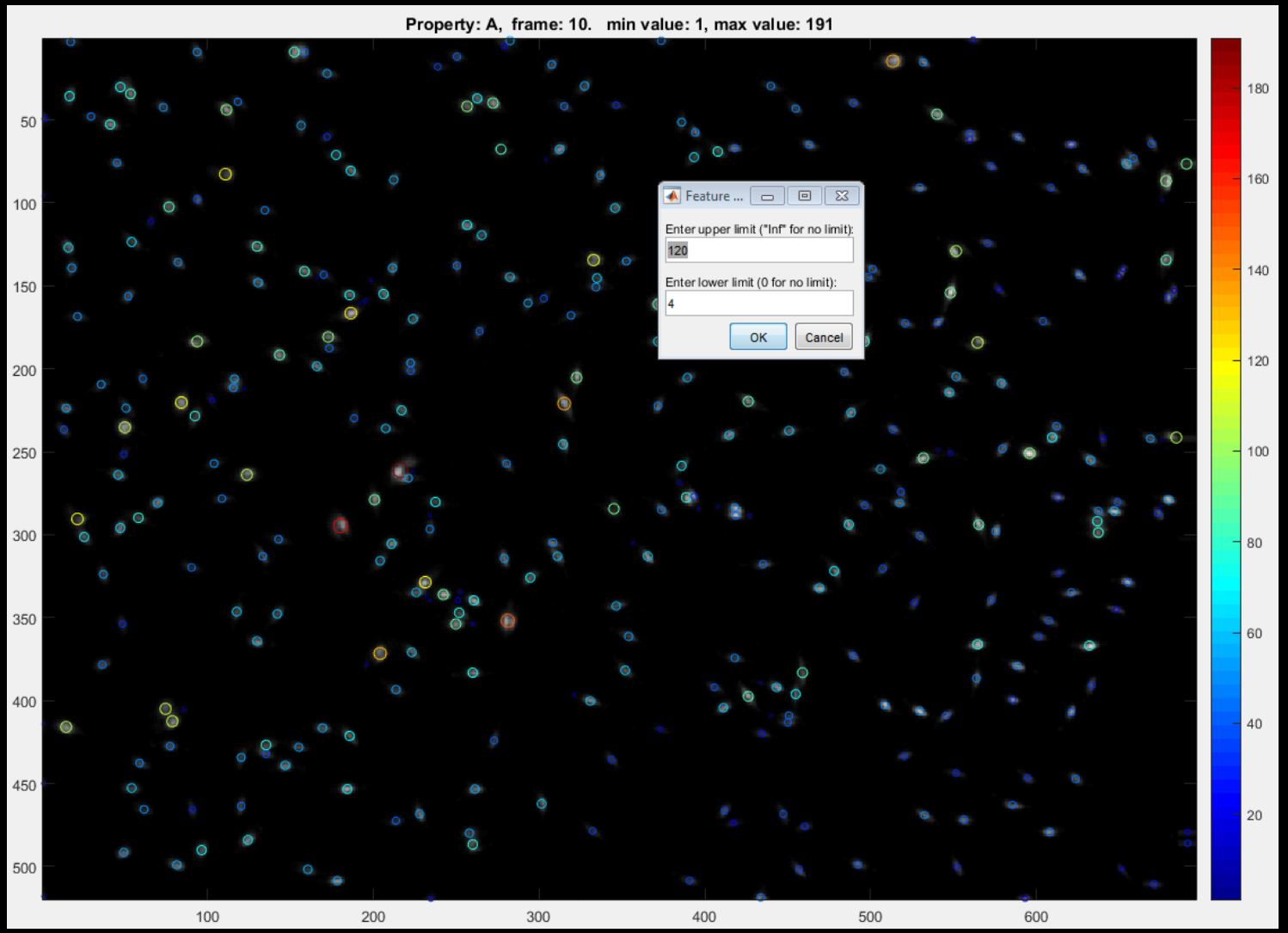


Cell properties and filtering

The feature is ranked, and the cells are marked with circles. The color represents the value for each cell. This is overlaid on a stack (in this case 405, but you can use other stacks as well).

Then, the values are verified with the user and can be updated.





Cell properties and filtering

The feature is ranked, and the cells are marked with circles. The color represents the value for each cell. This is overlaid on a stack (in this case BF, but you can use other stacks as well).

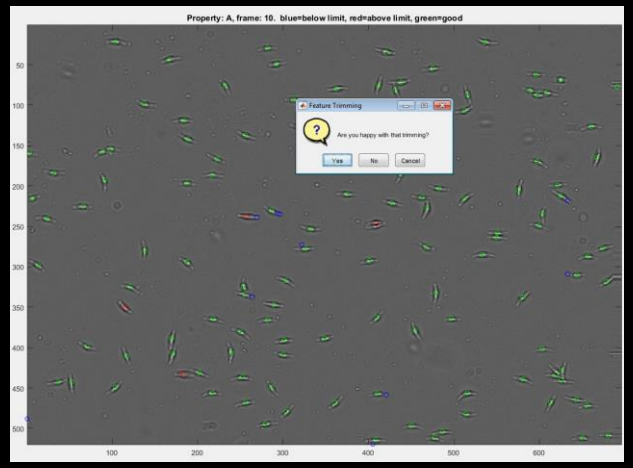
Then, the values are verified with the user and can be modified.

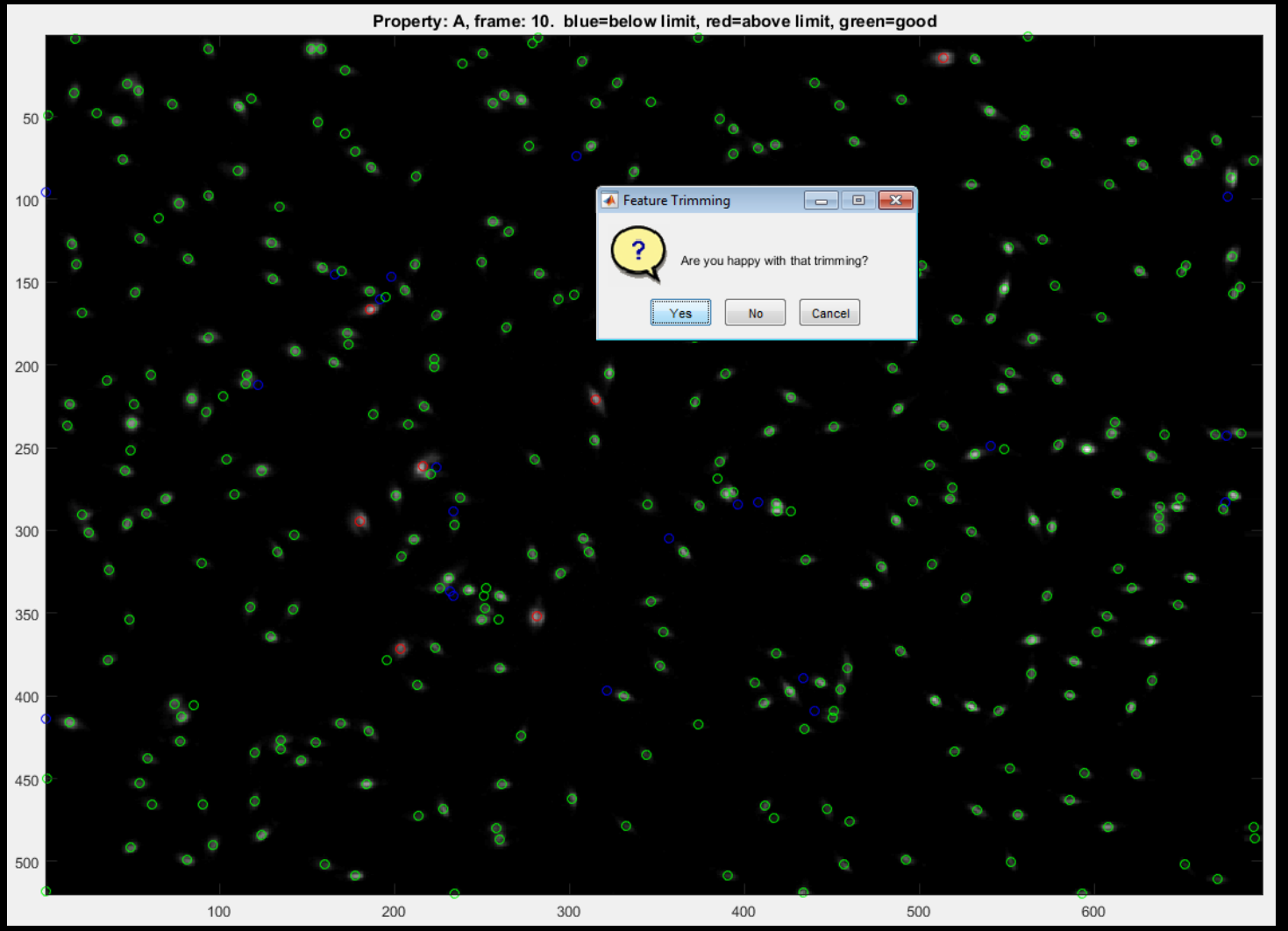
After that, the same image is shown with different color-coding of the circles:

Green – within the threshold.

Blue – too low.

Red – too high.



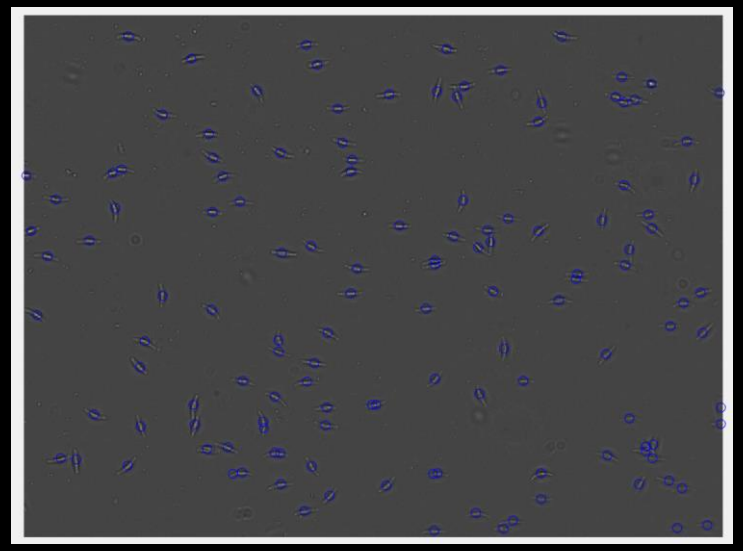


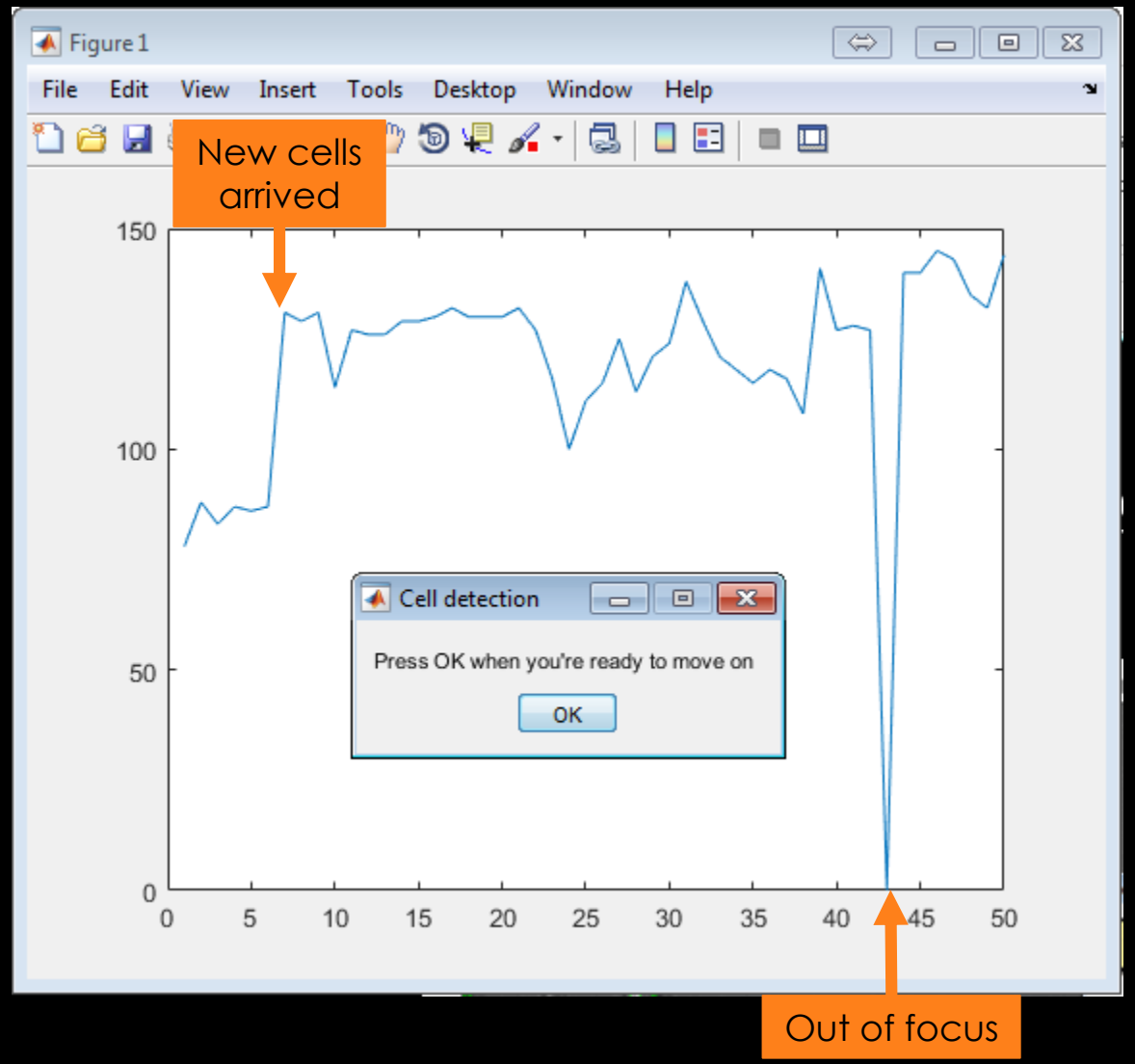
Cell properties and filtering

Before and after trimming the amount of cells detected in each frame is shown.

This can help detect out-of-focus frames or events in which large number of cells arrive or being washed away.

After filtering, you can view a movie of the detected cells over BF (with circles representing the cells).



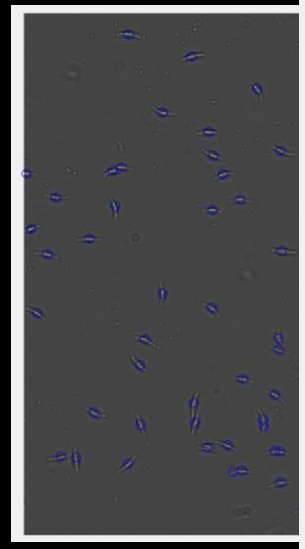


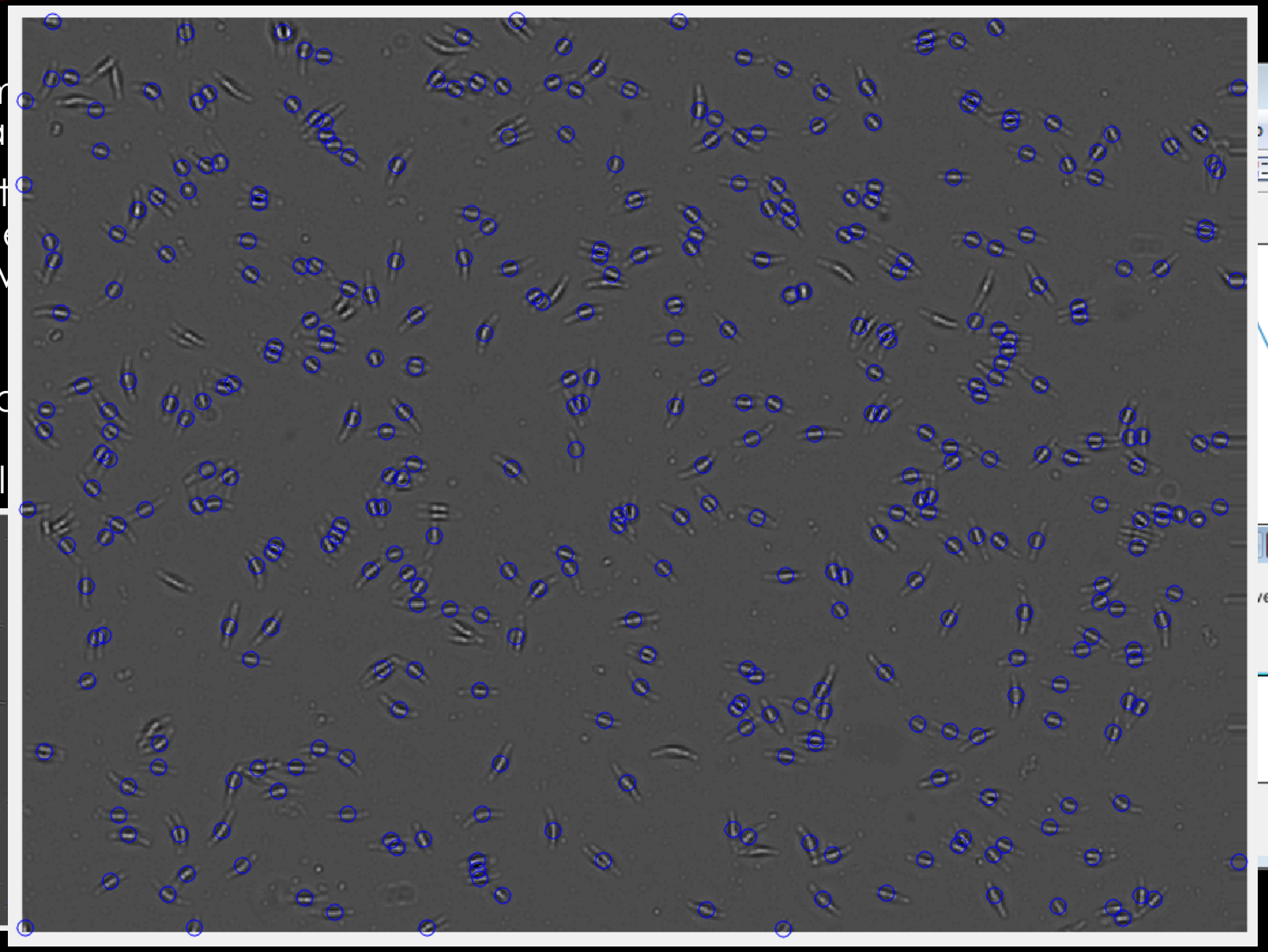
Cell properties and filtering

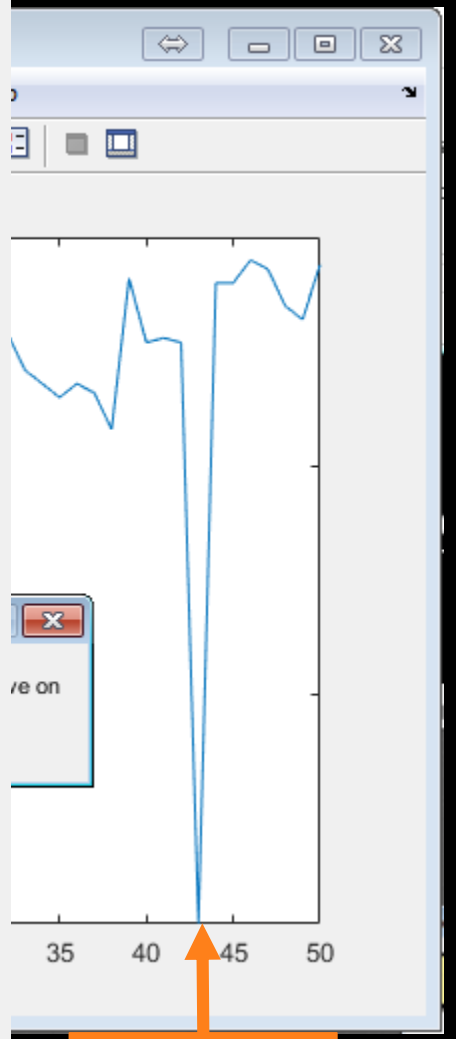
Before and after trim
cells detected in ea

This can help detect
events in which large
or being washed aw

After filtering, you co
detected cells over
representing the cel







Cell tracking

In this part, the aim is to connect the cells detected in the different frames so that the same cell can be tracked over time.

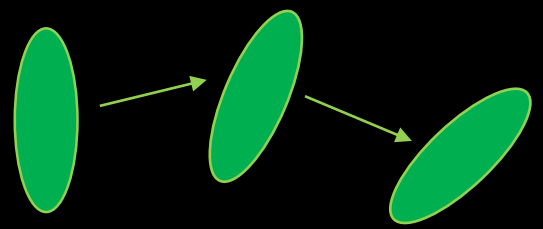
Vicente's tracking script has several modes (position, velocity and acceleration), but at the moment the Itria is using the simplest of them, which is **position** based. This is the mode that suits best to cells that do not swim (or don't move that much between frames).

The general idea is based on finding the **minimal distance** between the cell in one frame to the next frame.

Between each two sequential frames, all the distances between all the centers of cells in the first frame to those in the next frame are calculated. Then, the script finds the minimum distance and links these two cells as part of one track, with a specific "track ID". The two cells are removed from the matrix, and the next minimum distance is found, and so on.

Cells will be linked only if they are within a **distance limit** set by the user, which represents the maximum expected movement of a cell between two consecutive frames.

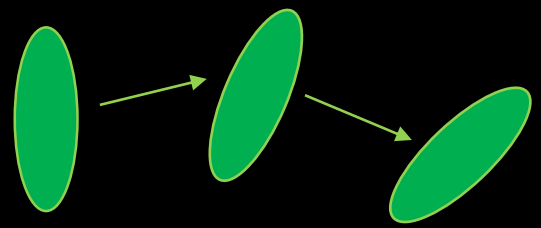
The distance is calculated between the **centers of the cells**, so the cell shape and other features are not taken into account.



Cell tracking

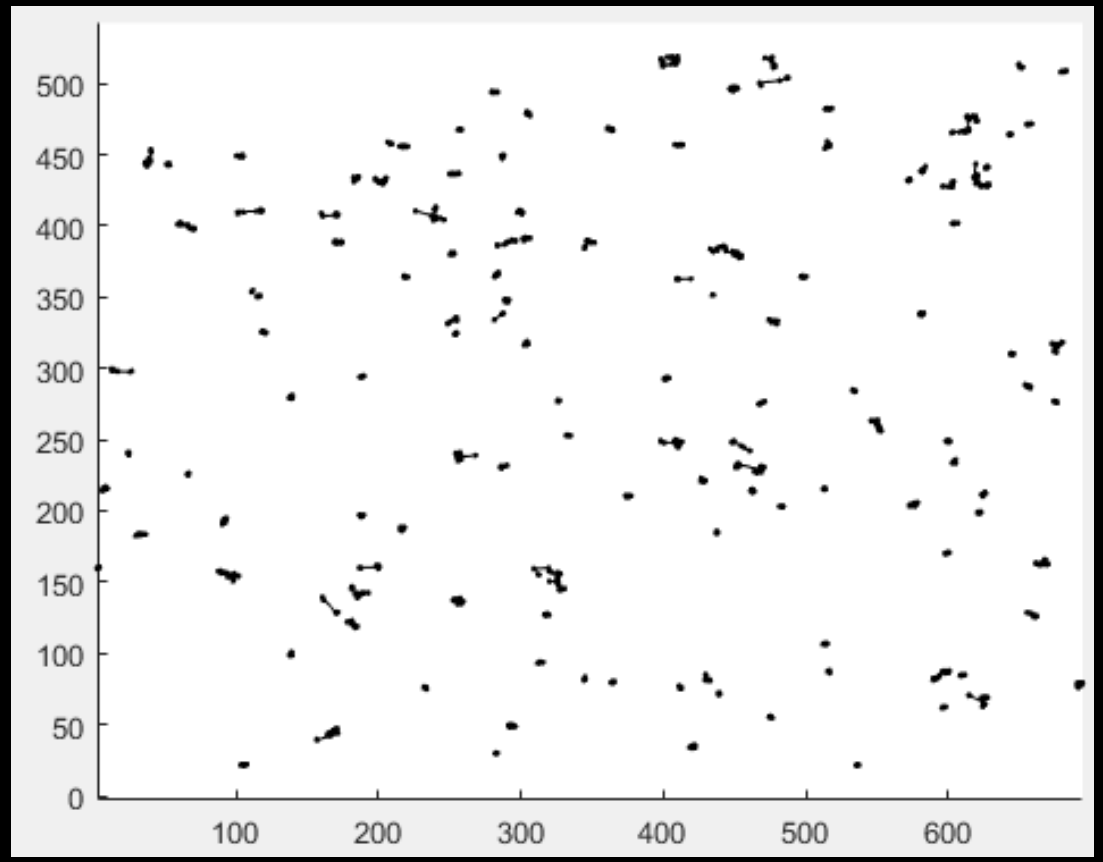
- After tracking, various features are being calculated such as velocity and acceleration.
- **Connect tracks with missing frames:** the script attempts to link tracks with missing frames (for example, if one frame was out of focus), based on the same distance limit that was used for tracking. The number of missing frames can be modified by the user, however it's best to keep it to the minimum in order to avoid false linking. At the moment it is set to 2 frames.
- **Track trimming:** remove tracks that are two short. At the moment, the minimum length is 6 frames.
- **Re-calculate track features:** after trimming and connecting the features are re-calculated.

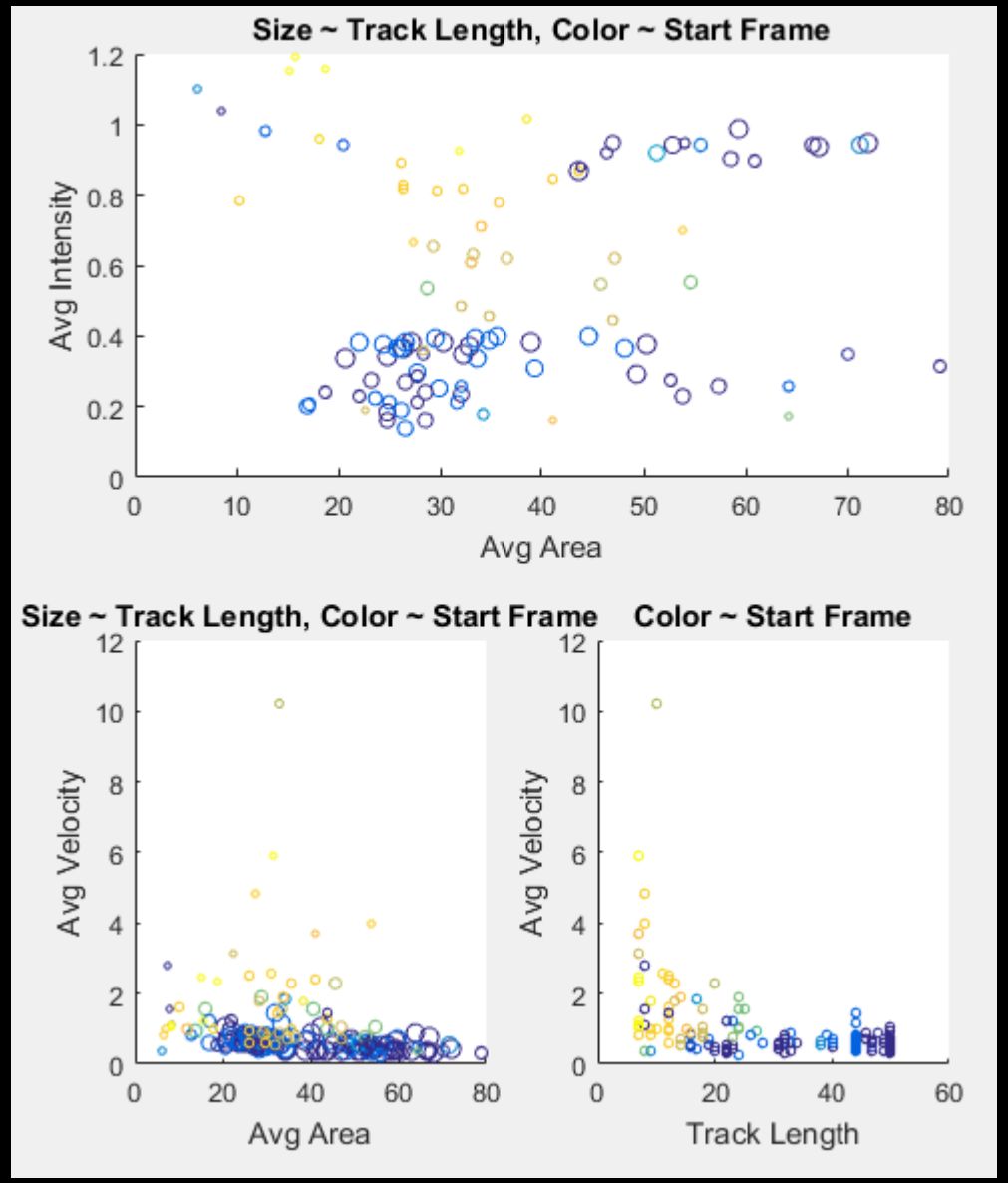
At the end you get two structures that contain the tracks and their features.
One is frame-based, and the other is track-based. This is for analysis purposes.



Cell tracking

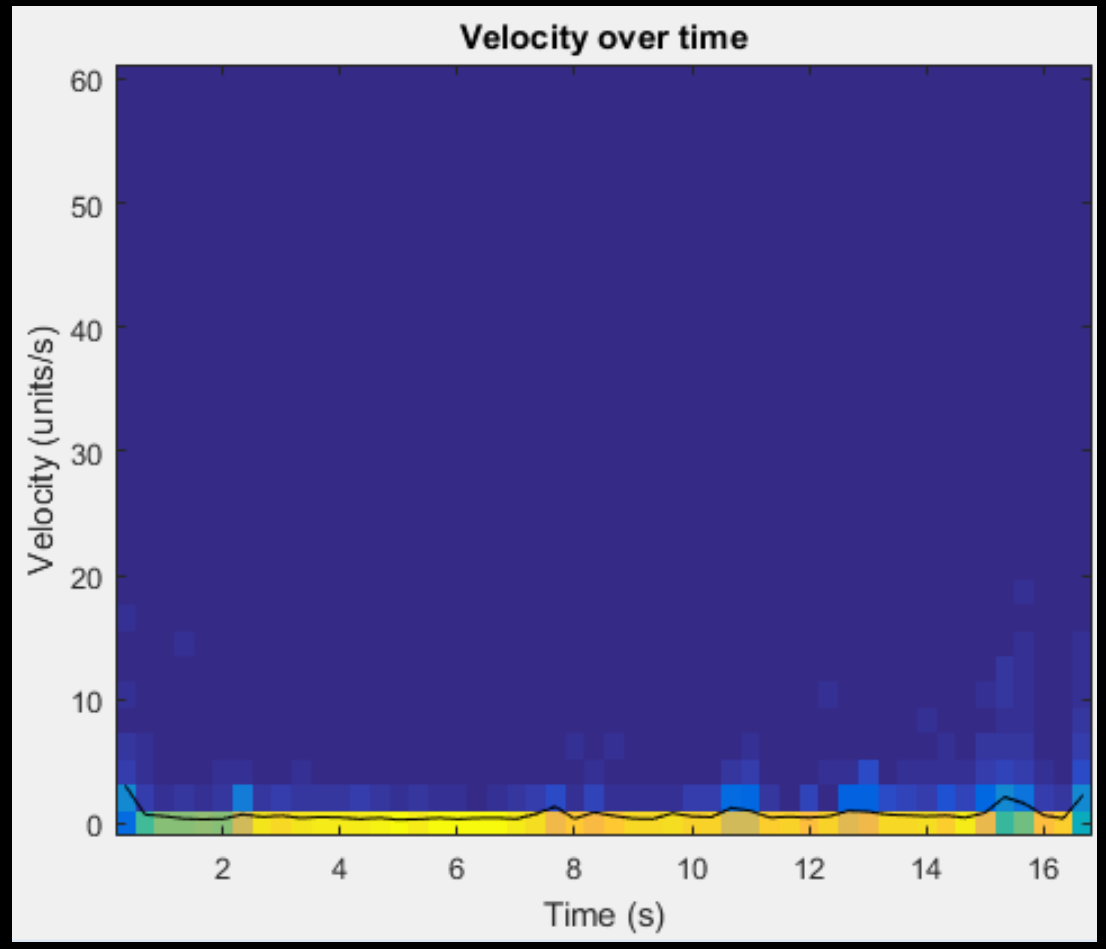
Several track statistics are viewed at the end, most of which are less relevant for the Itria purposes but it's still nice to have.

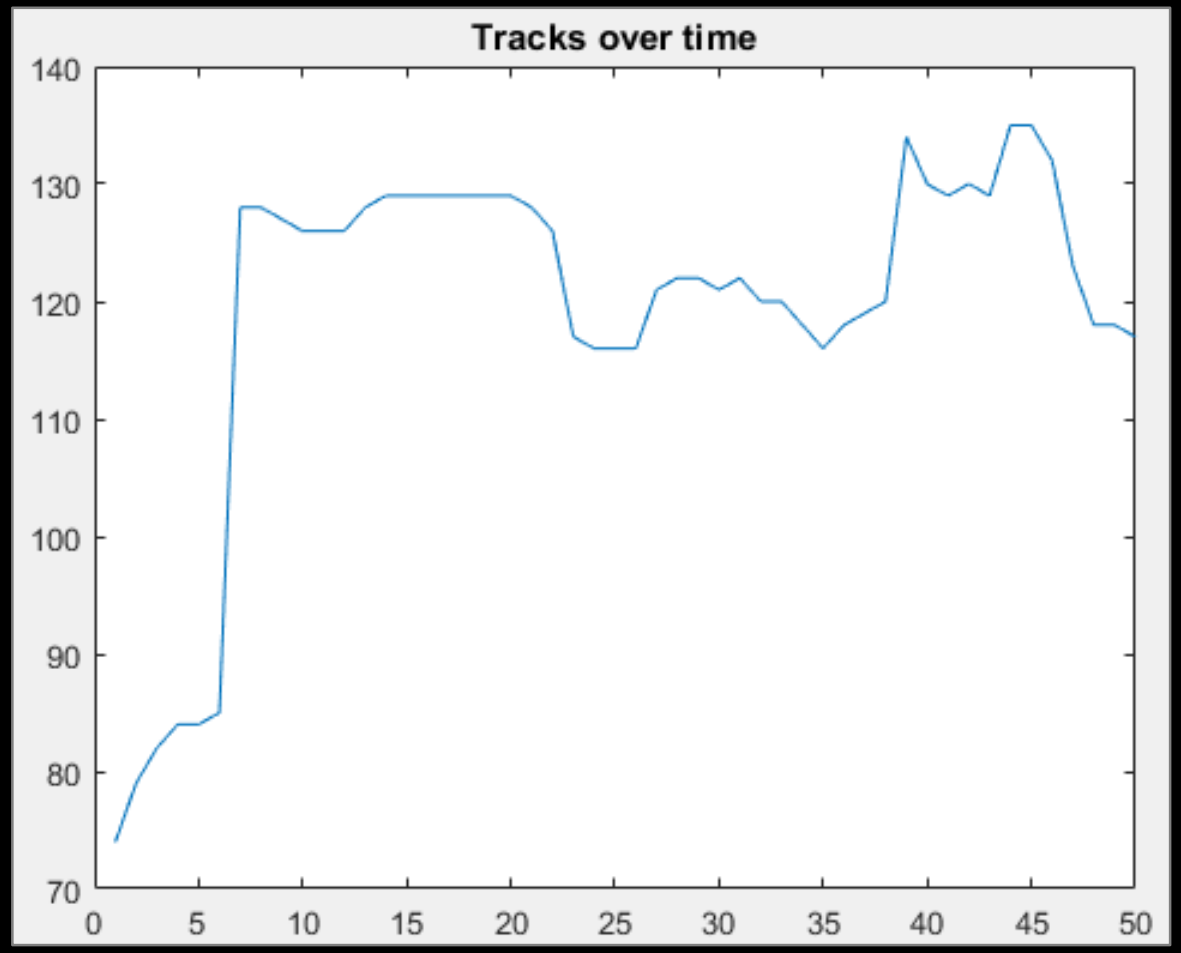




Cell tracking

Several track statistics are viewed at the end, most of which are less relevant for the Itria purposes but it's still nice to have.





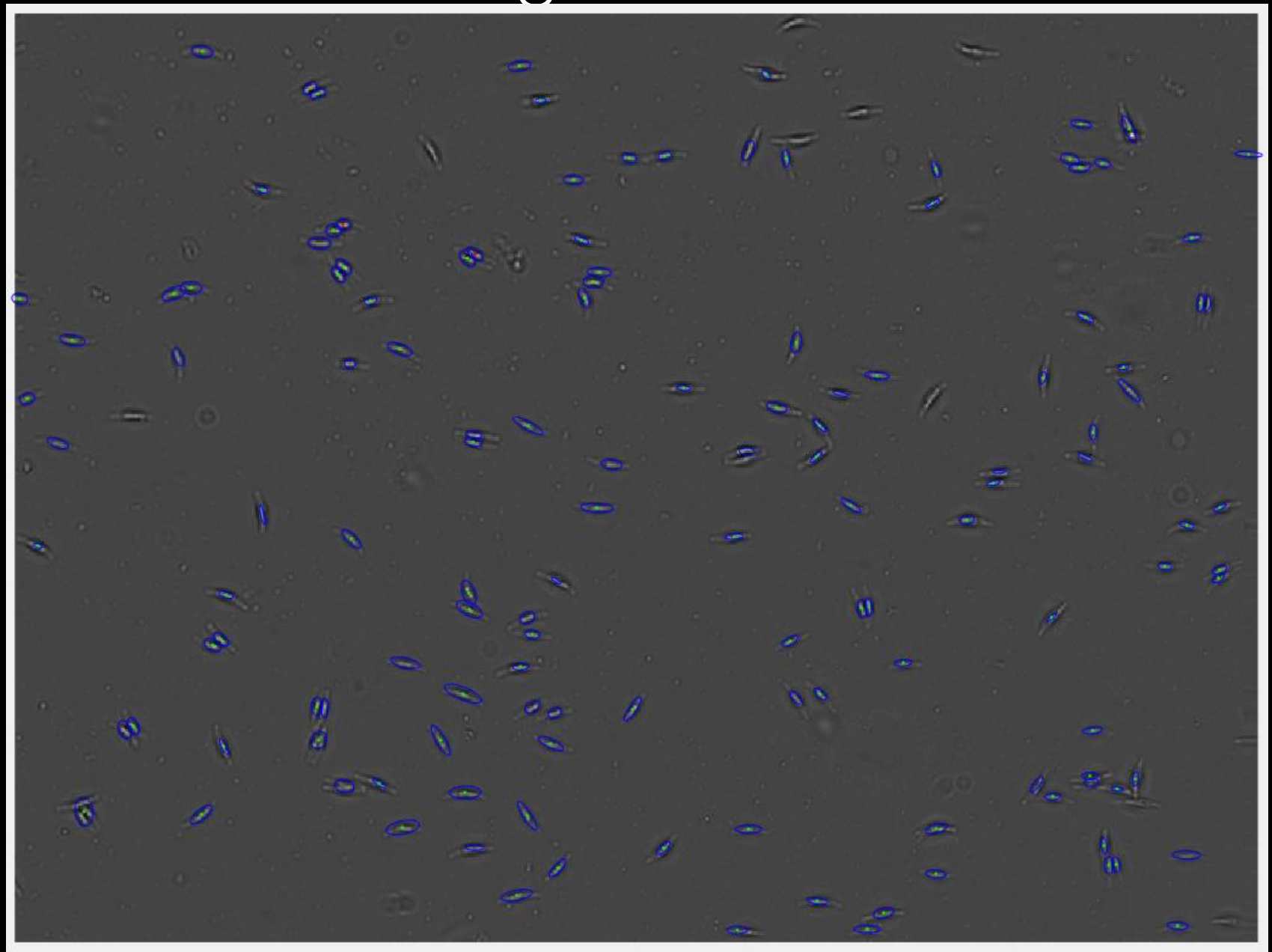
Cell tracking

View the final tracking:

Elipse with the major and minor axis length and orientation of the cell.

Red dot – ending track.

Green dot – new track.



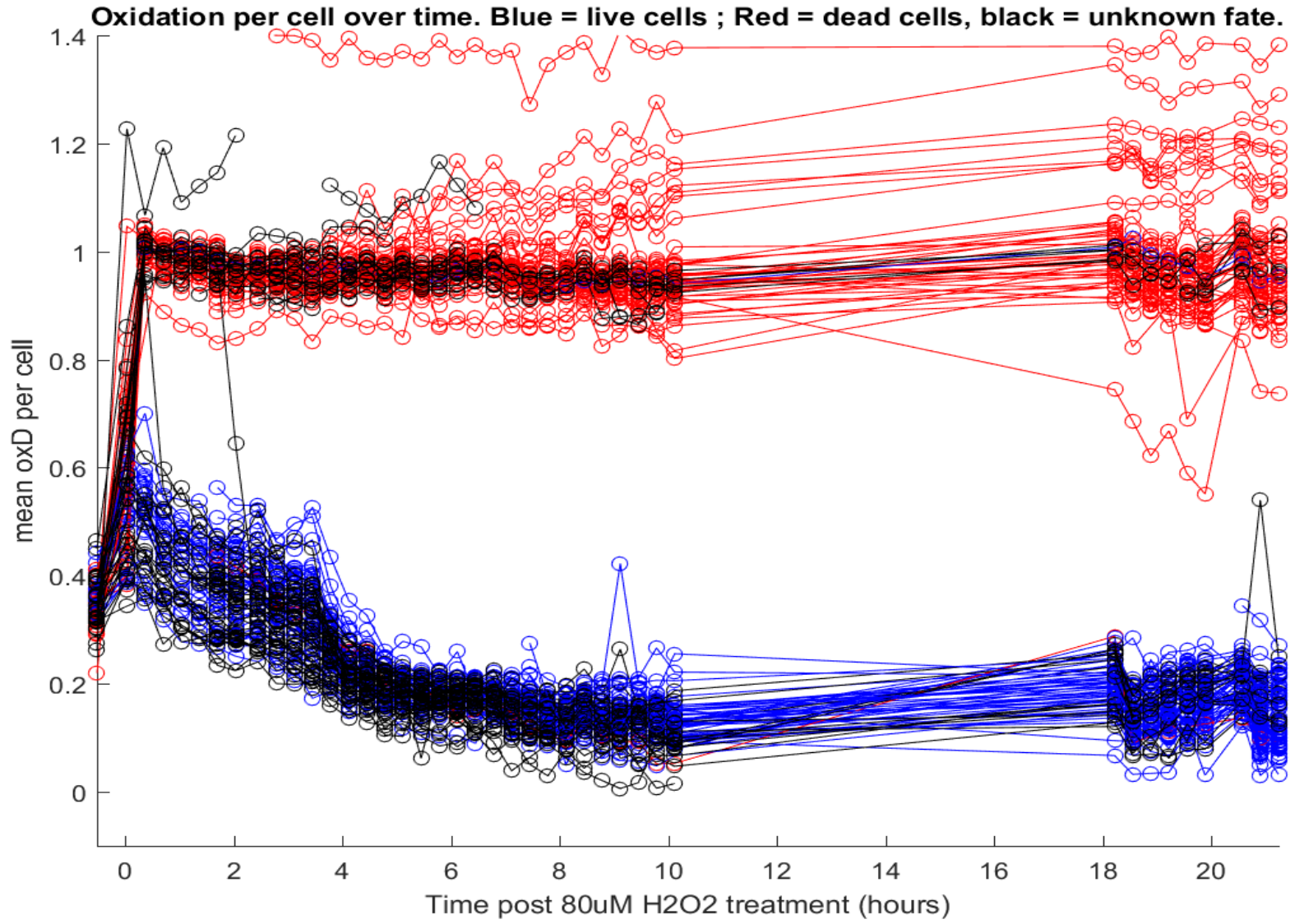
Here ends the part adapted from Vicente's script…

Analysis

- Add the time post treatment to the tracks features.
- Sytox analysis (as explained before).
- Calculate and plot oxidation over time per cell, with cell fate.
- Plot track trajectories (with or without cell fate)
- Add and plot other measurements (for example chlorophyyl intensity).
- Trim the final tracks: only full tracks, from first frame to Sytox frame with known cell fate.

Oxidation over time - example

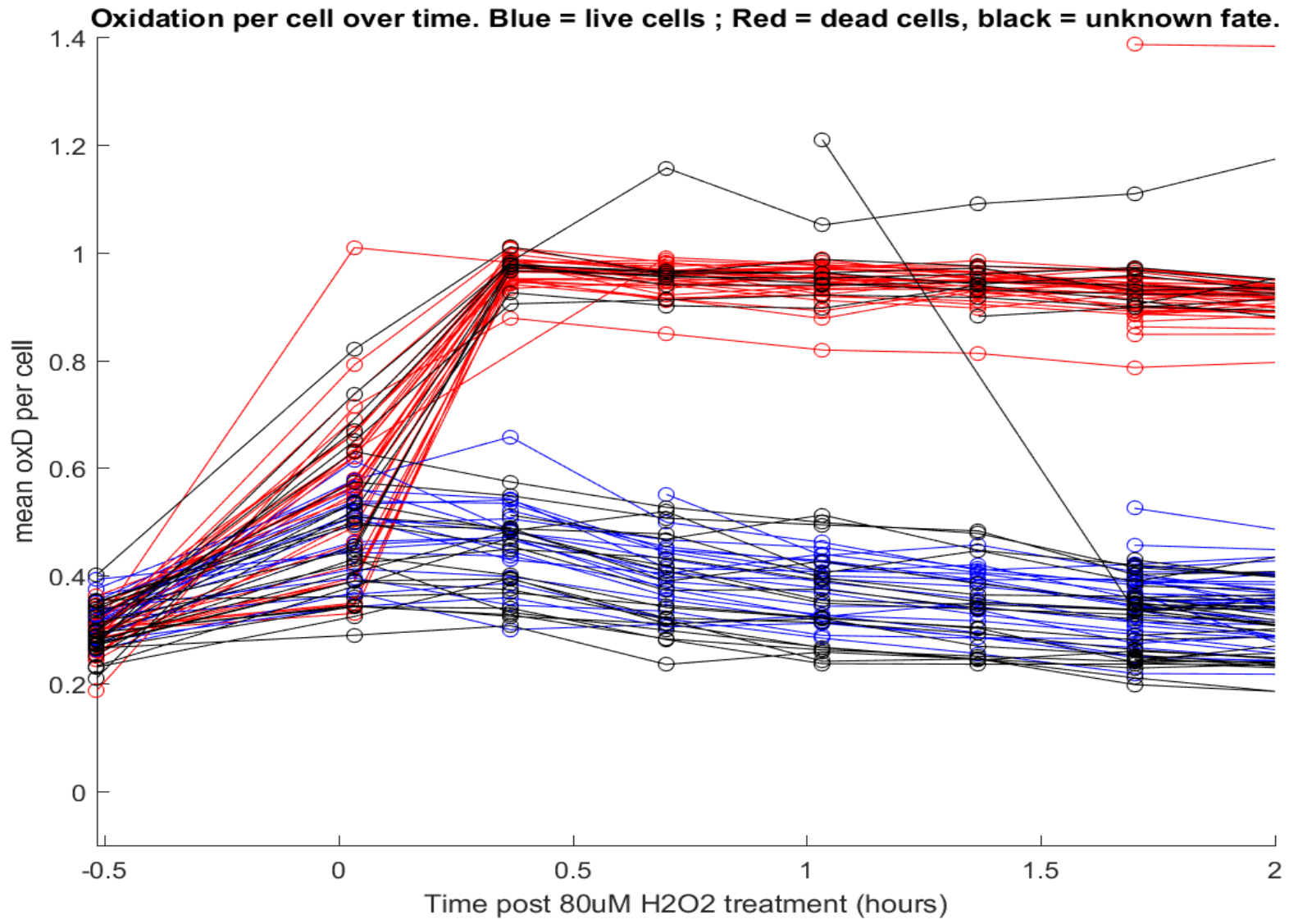
Field 12



*__reminder__: no imaging during the night

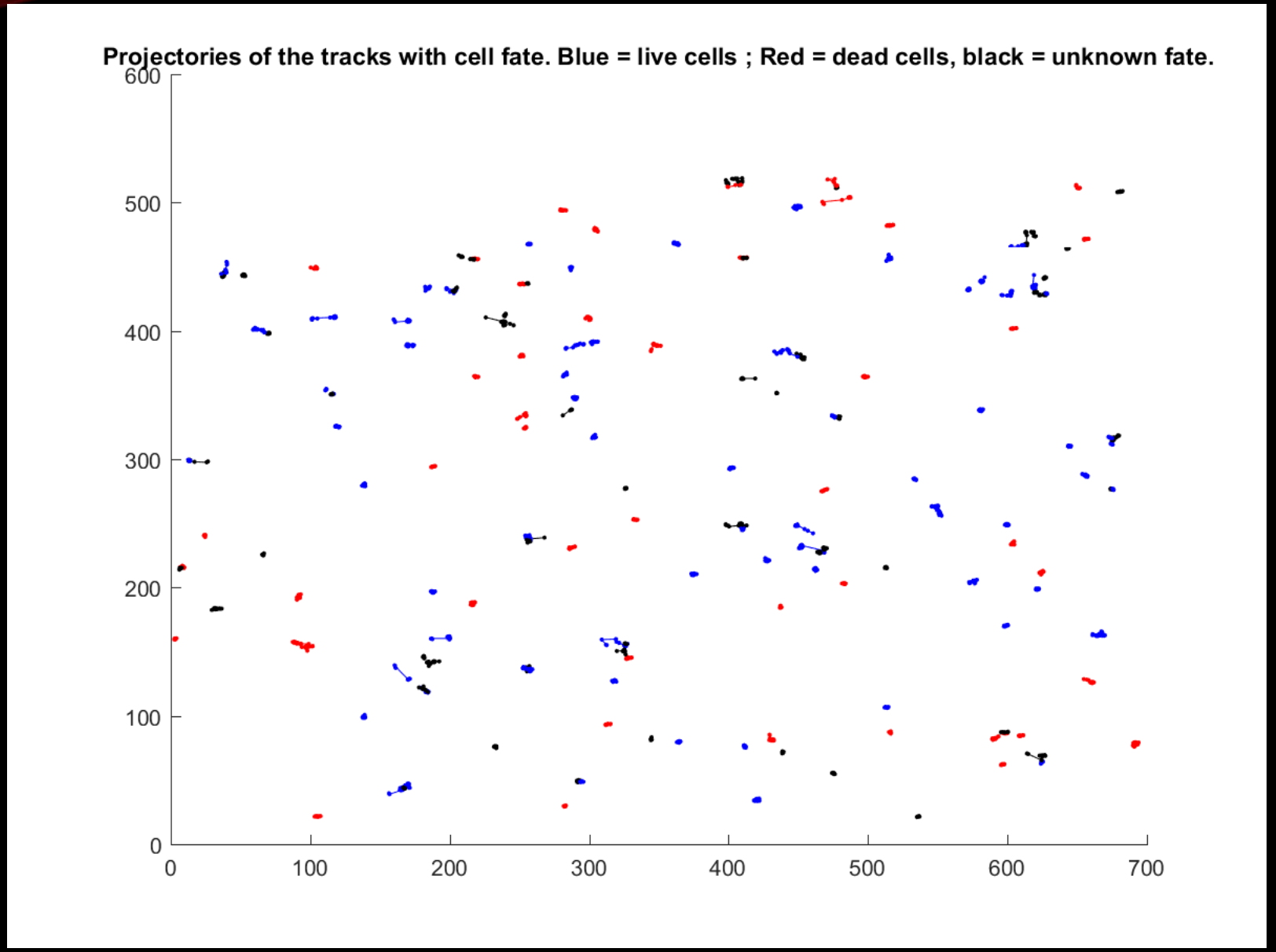
Oxidation over time – zoom in first 2 hr

Field 12

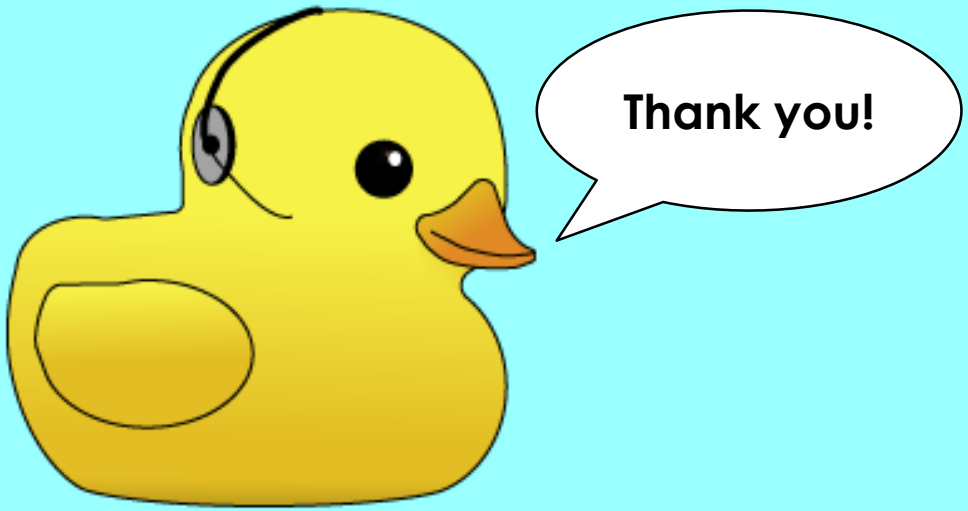


*__reminder__: no imaging during the night

Track trajectories example



Rubber Duck Debugging



Step 1) Beg, borrow, steal, buy, fabricate or otherwise obtain a rubber duck (bathtub variety)

Step 2) Place rubber duck on desk and inform it you are just going to go over some code with it, if that's all right.

Step 3) Explain to the duck what your code is supposed to do, and then go into detail and explain your code line by line

Step 4) At some point you will tell the duck what you are doing next and then realise that that is not in fact what you are actually doing. The duck will sit there serenely, happy in the knowledge that it has helped you on your way.

Note: In a pinch a coworker might be able to substitute for the duck, however, it is often prefered to confide mistakes to the duck instead of your coworker.

Original Credit: ~Andy from lists.ethernal.org

http://www.rubberduckdebugging.com/