User guide for Baxter Algorithms 1.6

Klas Magnusson
klasmagnus@gmail.com

Created: December 12, 2018

Contents

1 Introduction

The Baxter Algorithms (BA) is a software package for tracking and analysis of cells in microscope images. The software can handle images produced using either transmission microscopy (e.g. bright field, phase contrast, and differential interference contrast (DIC)) or fluorescence microscopy (e.g. wide field, confocal, and light sheet). The analysis of transmission microscopy images is limited to 2D, but 3D stacks of fluorescent images can be processed. In addition to cell tracking, the BA can perform automated analysis of fluorescent histological sections of muscle tissue, and automated analysis of myoblast fusion. The software is written in MATLAB, but it also contains some algorithms written in C++, which are compiled into mex-files.

The software as a whole is presented in [1] and an example of how the software can be used to analyze muscle stem cell (MuSC) behavior is found in [2]. The data association algorithm used to generate cell tracks is described in [3,4]. The software has shown outstanding performance compared to other software in the ISBI Cell Tracking Challenges of 2013, 2014, and 2015 [5–7].

2 Quick guide

I am aware that many users will not take the time to read this manual when they start using the software, and therefore I have put together a quick guide. It will be helpful when you start using the program and when you try to remember how to do things, but please try to read the entire manual at some point. It will save you time in the long run.

1. Download the software from `https://github.com/klasma/BaxterAlgorithms`.
2. Install the software, or run the source code in MATLAB.
3. Put your image sequences (or individual images) in separate folders and put these folders in an experiment folder.

4. Open the experiment folder in BA.

5. Load appropriate settings or enter them manually.

6. If you have acquired multiple channels, you need to specify the settings "channelNames" and "channelTags" to tell the program how to identify the different channels.

7. If you have multiple channels, you need to set the colors of the different fluorescent channels using the menu option **Settings/Set fluorescence display**. You need to select the advanced option in the level menu for this menu option to show up.

8. Edit the segmentation settings if necessary, using the menu option **Settings/Set segmentation parameters**.

9. Start the processing using the menu option **Automated/Track**. For transmission microscopy of cells in microwells, you first need to run **Automated/Stabilize**, and possibly also **Automated/Cut microwells**.

10. Review the tracking results using the menu option **Manual/Track Correction** and edit the tracks and/or the outlines if you think that it is necessary for the analysis that you want to do.

11. Analyze the tracking results. The graphical user interface (GUI) under **Analysis/Cell Analysis GUI** shows how properties of individual cells evolve over time, and the GUI under **Analysis/Population Analysis GUI** shows cell properties averaged over time.

12. Develop your own analysis scripts using MATLAB. The most important classes to know about are:

    - Cell
    - Blob
    - ImageData

    The most important functions to know about are:

    - LoadCells
    - SaveCells

    The functions are documented with comments in the files, and the documentation can be displayed by typing **doc** or **help** followed by the function name in the MATLAB command window.

3 Video tutorials

In the YouTube playlist `https://tinyurl.com/ba-tutorials` there are 16 video tutorials which give you step by step instructions on how to perform different tasks in the program. The topics of the tutorials are listed below.

1. Introduction
2. Installing and running the program
3. Image stabilization
4. Cutting of microwells
5. Opening multichannel data
6. Opening 3D data
7. Tracking in bright field microscopy
8. Tracking in fluorescence microscopy
9. Tracking in 3D fluorescence microscopy
10. Track correction
11. Segmentation correction
12. Data analysis
13. Writing analysis scripts
14. Fiber analysis
15. Myotube fusion analysis
16. Loading segmentations

4 Installation

The program is publicly available as a git repository on `https://github.com/klasma/BaxterAlgorithms`. The software is available in the form of source code and deployed applications for Windows and Mac. All forms can be downloaded by clicking on the **releases** tab and expanding the **Assets** dropdown under the latest release. If you have git installed, you can also clone the git repository to get access to the version controlled source code. The deployed software only runs on 64-bit Windows and Mac systems, but the source code be executed in MATLAB on other systems after compiling C++ components as described in Section 17.8.

4.1 Running the deployed application

If you do not have a MATLAB licence, you can run a deployed version of the program. The deployed versions can be found on the **releases** tab on https://github.com/klasma/BaxterAlgorithms. There, you expand the **Assets** dropdown under the latest release, and download the installer for your operating system. The installers for Windows and Mac are called *BaxterAlgorithmsInstaller_win64.exe* and *BaxterAlgorithmsInstaller_maci64.app* respectively. To run the deployed program, you need the MATLAB Compiler Runtime (MCR). The correct version of the MCR will be downloaded and installed during the BA installation, if it is not already installed. On Windows, text output will be written to a dedicated MS-DOS Command Window which is opened when you start the program by clicking on the icon on the desktop or in the start menu. On Mac, you can open the application by clicking on the icon, but to get text output, you need to start the program from the terminal as described in Section 6.1.

4.2 Running the source code

If you have MATLAB installed, you can execute the source code directly in MATLAB. The BA has been tested in MATLAB 2015b and 2018b. The software should work for those versions and all versions in between. Later versions are likely to work as well, but minor changes to the code may be necessary as MATLAB syntax is not always backward compatible. In addition to MATLAB, you also need the tool boxes for Image Processing, Optimization, Parallel Computing, and Statistics. All the necessary toolboxes are usually included in academic licences. If you have git, the easiest way to get the source code is to clone the git repository https://github.com/klasma/BaxterAlgorithms. Otherwise, you can download the source code from the repository website by clicking on the **releases** tab and expanding the **Assets** dropdown under the latest release. In MATLAB, you need to open the file *BaxterAlgorithms.m* and run it by pressing the run button in the MATLAB editor. When you press **run**, MATLAB might say that the file is not found in the current folder, and in that case you press the button to change folder. The first time you run the program, there will be a dialog saying that the program needs to modify a file called *javaclasspath.txt*.

5 Image data format

The program is designed to process 8 bit or 16 bit gray scale images. The program can also process color images with 3 RGB channels, but that is done by internally converting the color image to gray scale. The program recognizes the file extensions tif, tiff, png, jpg, and jpeg. I recommend that

you use tif-files, because most of the testing has been done with tif-files, and some read operations are faster when that file format is used. The images need to be organized into individual folders for the different image sequences (even if an image sequence only has a single image). These folders are then put into an experiment folder. The filenames in an image sequence folder do not have to follow any particular convention as long as the files are in the correct order when alphabetized. To ensure this, you need to pad numbers with zeros as in *image_02.tif*, so that *image_10.tif* is not put before *image_2.tif*. The program will save meta data and algorithm parameters associated with the image sequences in a file named *Settings.csv* in the experiment folder. Tracking results and other data created by the program will be stored in a folder named *Analysis* inside the experiment folder.

5.1 3D volumes

If you have 3D z-stacks, you can either save each z-stack as a tif-stack or save the z-slices as individual images. If you save individual images, you need to make sure that the alphabetical order of the files is such that the time points are in alphabetical order and such that the z-slices in each block of files are ordered from the first slice to the last slice. To make the program load the 3D volumes correctly, you need to specify the settings "numZ" and "zStacked" described in Table 1.

5.2 Multiple channels

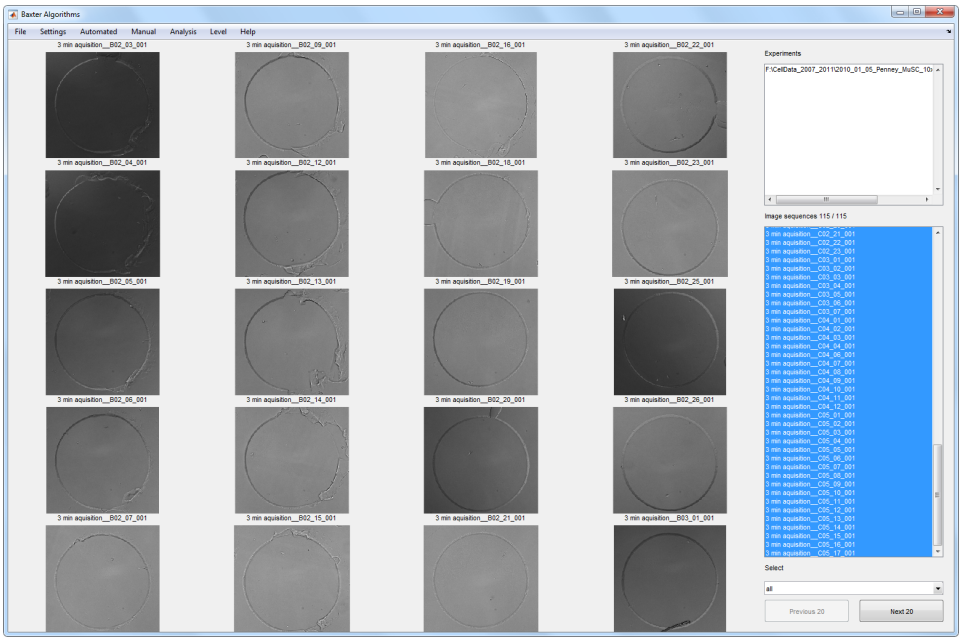
If multiple channels have been acquired, for example if there are fluorescent channels in addition to a transmission microscopy image, all channels should be put in the same image sequence folder. This requires that each channel has a part of the file name which is specific to that channel. To make the program identify the channels correctly, you need to set the "channelNames" and "channelTags" settings as described in Section 7.6.

6 Running the program

You start the program by either double clicking the icon of the deployed application or by running *BaxterAlgorithms.m* in MATLAB as described in Section 4.2. Starting the program opens the window shown in Figure 1.

6.1 Text output in the deployed application for Mac

During execution, the program will produce text output with progress indications, results, and error messages. When the source code is executed in MATLAB, the output is sent to the Command Window. When the deployed application is executed on Windows, the the output is sent to an MS-DOS



Credit: *Penney Gilbert, Stanford University*

Figure 1: The main GUI window, from which all other GUIs can be opened.

Command Window which is opened together with the program. When the program is executed on Mac, you will not get any text output if you start the program by clicking on the icon. In order to get text output on Mac, you need to start the program from the terminal, using a shell script which is located inside the .app-folder associated with the program. The shell script takes the path of the installed Matlab Compiler Runtime (MCR) as an input argument. To start the program in this way, you first open a terminal window. One way to do that is to click the magnifying glass in the upper right corner of the screen, and typing **Terminal** into the search field. You run the shell script by entering the full path of the shell script followed by the full path of the MCR. If you installed both the BA and the MCR in the default locations, you enter **/Applications/BaxterAlgorithms/application/run_BaxterAlgorithms.sh /Applications/MATLAB/MATLAB_Compiler_Runtime/v95/**.

6.2 Opening experiments

You open an experiment by clicking on **File/Open experiment** and selecting the desired experiment folder. If you have a single image sequence stored in a folder, you must put this folder into an experiment folder and open that folder in the program. You can open multiple experiment folders

by either selecting multiple folders in the experiment opening dialog or by clicking **File/Add experiment**. Opening new experiments using **File/ Open experiment** will automatically close all open experiments.

6.3 Browsing through image sequences

Once an experiment is open, you can see the first images of the image sequences by selecting their file names in the list box labeled **Image sequences**. Different subsets of the image sequences can be selected using the dropdown menu labeled **Select**. If more than 20 image sequences are selected, you can browse through them using the buttons **Next 20** and **Previous 20**.

6.4 Performing tasks

All tasks that can be performed on the image sequences can be found in the menus of the main window. Clicking such a menu will in general perform the corresponding task on the image sequences that have been selected.

6.5 User levels

The menus and the settings of the program have been classified into the three different levels "basic", "advanced", and "development". This has been done to guide users to the most important functions and settings, and to hide some of the more advanced functionalities from users how are learning the software. You can select a user level in the **Level** menu in the main GUI. The GUIs found under **Settings/Settings** and **Setting/Set segmentation parameters** also have **Level** menus, and they are independent from the one in the main GUI.

The basic level has all the important functions and settings necessary to process image sequences. The advanced level has additional functions and settings which provide higher performance and flexibility. The development level has functions that developers may find interesting. These functions and settings are either experimental or have very limited utility. Functionalities at the development level may also not work properly in the current version of the program, and may be removed in a future version. Avoid using these functionalities if you do not need them.

6.6 Players

Many of the functions in the menus will open different players, where the image sequences can be played and processed or analyzed. These players all share the same structure, but they have different buttons and menu options. The simplest player is found under **Manual/Play** and only plays the image sequences. Before any analysis is done, it is good to look at the

image sequences in this player to make sure that all the image sequences are saved in the correct format and that all image settings are correct.

6.6.1 Zooming and panning

All of the players that show images allow you to zoom and pan. You zoom by holding down the left mouse button somewhere in the image and dragging to create a rectangle around the region that you want to display. You zoom out by right-clicking. When you have zoomed in, you can pan around in the image by holding down the space bar and dragging with the left mouse button down. You can also toggle panning by pressing **m**. You press **m** once to enable panning and then again to disable it. That can be useful if you are working on a laptop which disables the touchpad when you hold down space. In the players for manual correction, zooming is turned on by selecting the zoom tool, but the panning tool is always turned on so that you can move around in the image quickly. In other players, zooming and panning are always turned on.

6.7 Visualizing 3D data

When you view 3D data in a player, as in Figure 2, an additional control panel will appear to the right of the image. That control panel has controls which determine how the 3D data is displayed. In the **display** dropdown menu, you can select what 3D views you want to look at. The default is to show the xy-view, but you can also show the xz-view or the yz-view, or all views at once as in Figure 2. By default, the players will show maximum intensity projections through the 3D volume, but you can also select to show individual slices by un-checking the checkboxes **x proj.**, **y proj.**, and **z proj.**. Once you are displaying individual slices, you can select which slice to display using the text boxes and sliders labeled **x**, **y** and **z**. You can also select slices in the images by shift-clicking or clicking with the middle mouse button, in one of the other 3D views. Lines in the other 3D views show the locations of the selected slices. Controls which have no effect on the current visualization are disabled. You can zoom and pan in the different 3D views, just as in 2D data. When you have zoomed in, maximum intensity projections will be computed over the zoomed in volume and not over the whole z-stack.

6.7.1 Displaying multiple channels

By default, the players will display all channels as a single merged image, but under the menu option **Channels**, you can select what channels you want to include in the merge. The image sequence will play faster if fewer channels are selected, especially if only a single transmission microscopy channel is

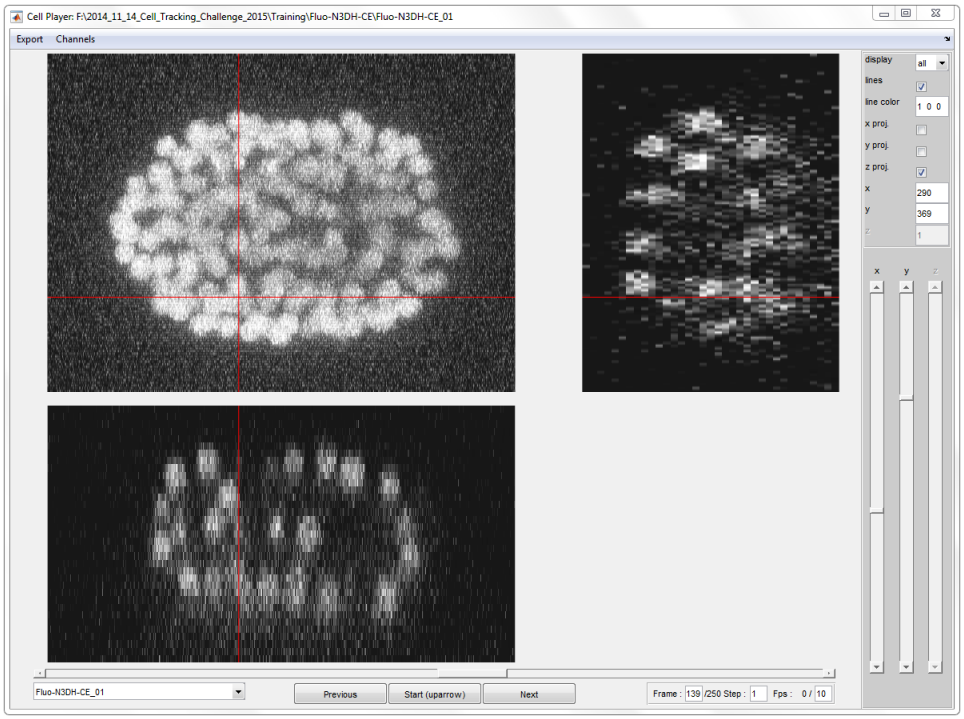


Figure 2: Controls which are used to change how 3D data is displayed.

selected. Details about settings for the display of multiple channels can be found in Section 7.6.

6.7.2 Exporting images

All of the players can export the currently displayed image to an image file using **Export/Export image**. In general, all of the images and plots displayed in the figure will be included in the saved image, but none of the user controls will be included. The default is to save a tif-image, but you can save a different file format by specifying the appropriate file extension in the filename selection dialog.

6.7.3 Exporting videos

In the players it is also possible to export image sequences using the menu options **Export/Export image sequence** and **Export/Export all image sequences**. The program plays the image sequences and uses screen capture to generate individual png-images for all time points. The sequences of png-images can then be converted into video files using third party software, such as ImageJ or VirtualDub.

7 Settings

There are a lot of settings that affect different parts of the data processing. A short explanation of the settings can usually be displayed by holding the mouse cursor over the setting. The program also has predefined settings for different types of image data.

7.1 csv-files

Settings are saved in csv-files named *Settings.csv* in the experiment folders. The csv-files are comma separated text files, so you cannot use commas in any of the settings. The files have a row for each setting and a column for each image sequence, and can be opened and edited in text editors or in Excel. Excel can however make unexpected changes to settings values if the program makes incorrect assumptions about data types.

7.2 Loading settings

You can load settings by pressing either **Settings/Load Settings** or **Settings/Load Settings (browse for file)**. Pressing **Settings/Load Settings** opens the GUI shown in Figure 3, where you can open all of the settings files that are distributed with the program. The GUI shows information about what types of images the settings can be used for, and displays a sample image from a dataset that has be processed using the settings. Pressing **Settings/Load Settings (browse for file)** opens a file selection dialog where you can browse for a settings file. This lets you load settings files that you have saved, settings files from other datasets, or settings files that the program generated when you ran tracking on an image sequence. Whenever you run tracking, the program will save a settings file with all of the settings that were used, in the same folder as the tracking results. This is useful when you want to reproduce your results.

When you have selected a settings file to load, you can choose to load all of the settings saved in the file, or a subset of them. If there are settings for multiple image sequences in the saved file, you can choose what image sequence to load settings from. You can either pick one image sequence name for each image sequence that has been selected in the main GUI or choose a single image sequence name, from which settings will be loaded for all of the image sequences selected in the main GUI. When settings are loaded, the previous settings values will be overwritten, so you need to make a copy of the old settings file if you want to be able to go back to it. There are predefined settings files for tracking of cells in transmission microscopy, tracking of cells in fluorescence microscopy, tracking of MuSCs and myoblasts in bright field microscopy and for segmentation of fluorescently stained muscle histology sections.

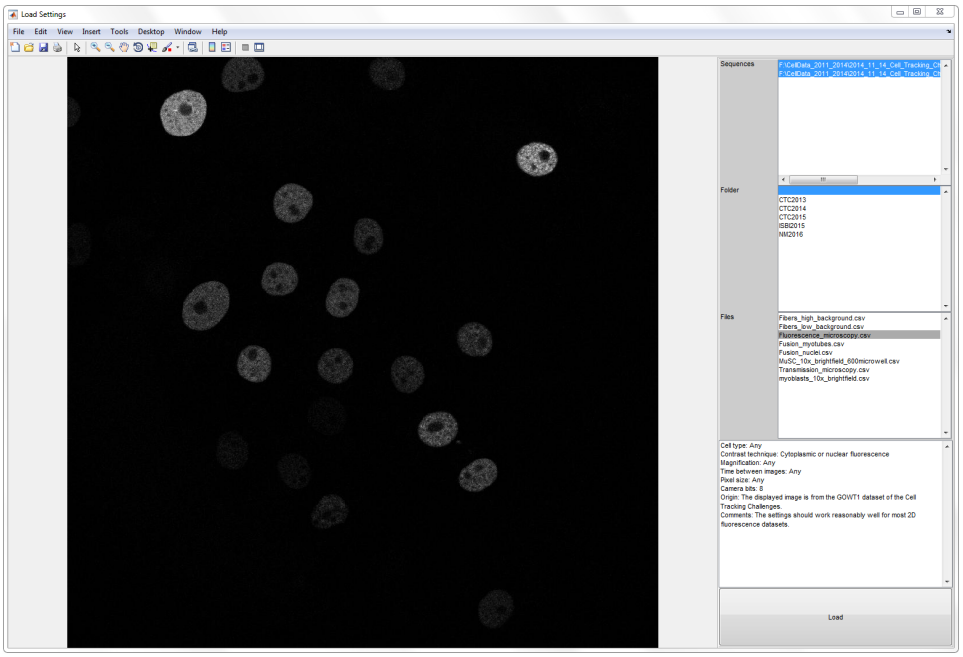


Figure 3: GUI where you can load settings files.

7.3 Saving settings

Settings can be saved by pressing **Settings/Save Settings**. This will save settings only for the image sequences that have been selected in the main GUI.

7.4 Settings GUI

The settings can be modified in the GUI shown in Figure 4 by pressing **Settings/Settings**. The settings are grouped into the four categories "image", "segmentation", "tracking", and "analysis", which can be selected in the **Category** menu. The image category contains settings which tell the program in what format the images have been saved and how they should be displayed. The segmentation settings tell the program how to find the outlines of the cells, and the tracking settings tell the program how to link the outlines into tracks. The analysis settings describe how the final tracking results should be visualized and analyzed. Many of the settings are binary and take the values 1 or 0 indicating "on" or "off" respectively. All algorithm parameters are given in pixels and frames. Settings which have no effect at the moment are not displayed, and therefore some settings may appear or disappear when the values for other settings are changed.

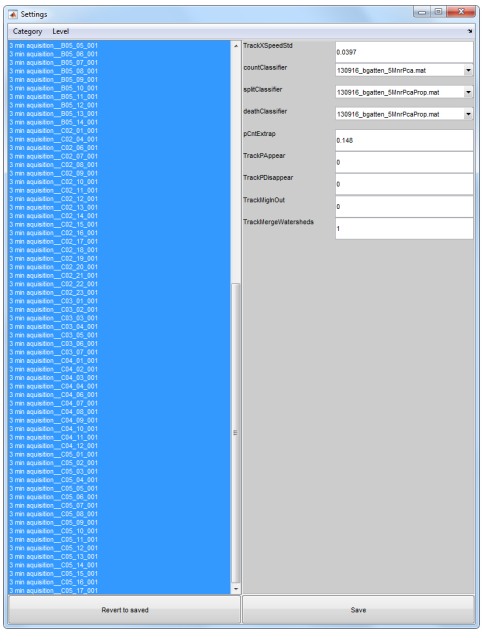


Figure 4: GUI where you can change settings associated with the image sequences.

7.5 Image settings

The settings in the image category are explained in Table 1. On the basic level, you can only change the number of image bits used by the camera. To access the other settings, you need to select the advanced level. The advanced settings need to be specified correctly in order for 3D data and data with multiple channels to display properly.

7.6 Fluorescence settings

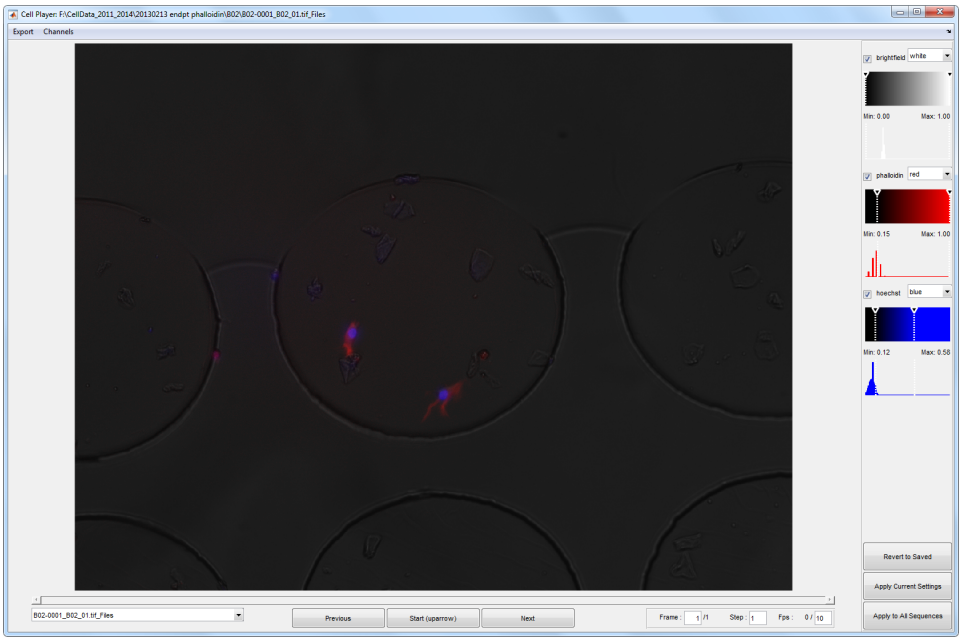
If the data contains multiple channels, you have to tell the program what the channels should be called and how images from the different channels should be identified. The names of the channels should be entered as text strings separated by ":" in the field "channelNames", and character sequences that are specific to images from the different channels should be entered as text strings separated by ":" in the field "channelTags". If for example there is a bright field channel and a Green Fluorescent Protein (GFP) channel, the file names of which end with "c01" and "c02" respectively, you can enter "bright field:GFP" into "channelNames" and "c01:c02" into "channelTags".

Table 1: Settings associated with the image files.

use	If this setting is 0, the image sequence will not be processed or included in any analysis.
minWellR	Lower bound in the interval of possible microwell radii in pixels. Set this to "nan" if there are no circular microwells.
maxWellR	Upper bound in the interval of possible microwell radii in pixels. Set this to "nan" if there are no circular microwells.
channelNames	Names of the imaged channels, such as "bright field" or "GFP", separated by ":".
channelTags	Unique identifiers, such as "c01" and "c02", for files belonging to the different channels, separated by ":".
sequenceLength	The maximum number of images that will be analyzed in the image sequence. If this setting is left empty, all images are analyzed.
numZ	The number of z-planes in 3D data. This should be 1 for 2D data.
zStacked	Set this parameter to 1 if the files are tif-stacks with multiple z-planes in each file.
bits	The number of bits used by the camera. If the images have 8 bits, the number of camera bits is almost always 8, but if the images have 16 bits, the camera will often only use 10, 12, or 14 of them.
voxelHeight	The ratio between the voxel height and the voxel width in 3D data. The value is not used for 2D data.

7.6.1 Specifying channel colors

Once you have entered the names and the tags of the channels correctly, you can specify the colors for the different channels by clicking on **Settings/Set fluorescence display**. That opens the GUI shown in Figure 5, where you can specify colors and ranges of fluorescence values in the different channels.



Credit: *Ermelinda Porpiglia, Stanford University*

Figure 5: GUI where the colors and ranges for different fluorescence channels can be specified.

8 Stabilization

Image stabilization removes camera shake, caused by movements of the stage, by aligning the images in an image sequence. Image stabilization is performed by pressing **Automated/Stabilize**. This brings up a GUI where you can specify the path for the stabilized experiment, and specify how many image sequences should be stabilized in parallel. Performing stabilization creates a new experiment folder with stabilized image sequences, and leaves the original data unaltered. The **Queue** button lets you put the stabilization job in a queue, as described in Section 11.1. The image sequences are aligned using a MATLAB implementation of the image stabilization plugin for ImageJ [8]. The MATLAB code is less advanced than the ImageJ plugin in that it can only correct for translations in the x- and y-dimensions. It does however have a few additional features, such as the option to crop the field of view so that no unknown pixel values are included. The implementation has GPU support and will run about 3 times faster if the computer has a GPU.

9 Cutting of microwells

Clicking **Automated/Cut microwells** brings up a GUI which cuts out circular microwells from all images in the image sequences and saves the cropped images in a new experiment folder. As for stabilization, you can specify the path for the new experiment, and how many image sequences should be cut in parallel. The original data is unaltered. When both stabilization and cutting is performed, it is best to perform stabilization first, as pixels are lost around the image borders in the stabilization step. The **Queue** button lets you put the cutting job in a queue, as described in Section 11.1.

10 Segmentation

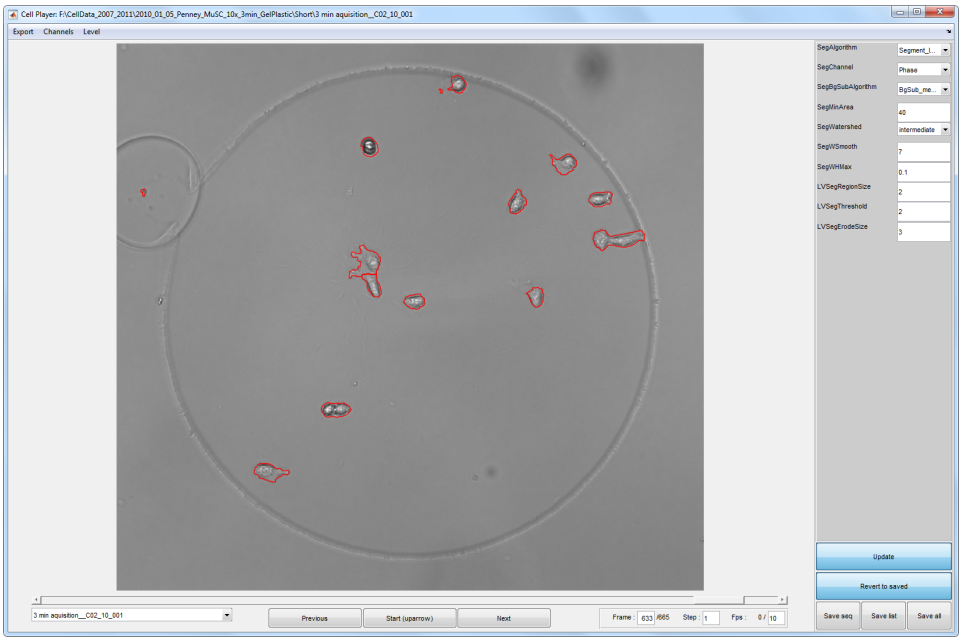
The first step in analyzing a dataset is to specify a segmentation algorithm that can find the cells in the images, and settings for that segmentation algorithm. Clicking on **Settings/Set segmentation parameters** opens the user interface shown in Figure 6, where the cell segmentation is displayed next to a control panel where the segmentation settings can be adjusted. There are many different segmentation algorithms to choose from and the most important ones are described in Section 10.2. In addition to the segmentation algorithms, you can select preprocessing algorithms that will be applied before the segmentation, and post-processing algorithms that will be applied after the segmentation. These algorithms are independent of the segmentation algorithm and are described in sections 10.3 and 10.5 respectively.

10.1 Segmentation GUI

The GUI has the player structure described in Section 6.6, so that different time points and different channels can be displayed easily. It is advisable to look at the segmentation results for multiple image sequences and multiple time points before starting the tracking, to make sure that the selected segmentation settings work for all images. One way to quickly look at multiple time points is to enter "10" into the **step** text box in the lower right corner of the interface, to display every 10th image in the sequence, and then play the sequence from beginning to end.

10.1.1 Choosing a segmentation channel

Before a segmentation algorithm can be chosen, you must select what channel the segmentation should be applied to, in the dropdown menu "SegChannel". Channels with nuclear fluorescence are usually easier to segment than channels with cytoplasmic fluorescence which are in turn easier to segment



Credit: *Penney Gilbert, Stanford University*

Figure 6: GUI where you can set segmentation parameters and preview the segmentation results before you start processing the image sequences.

than transmission microscopy channels. Section 7.6 has information about how to specify what channels an image sequence has.

10.1.2 Reusing an old segmentation

If you are happy with the segmentation in a generated tracking result, you can reuse that segmentation when you rerun the tracking. This can be useful if you have found good segmentation settings but need to change other settings to improve the tracking results. To reuse an old segmentation, you select the name of the corresponding tracking version in the dropdown menu "SegOldVersion" in the segmentation GUI.

10.1.3 Importing segmentations from other software

It is possible to generate segmentations using other software (such as CellProfiler or ImageJ) and use them for tracking. To do this, you need to save the segmentations of the individual images as 16-bit tif-files, with label images where the background pixels are 0, the pixels of region 1 are 1, the pixels of region 2 are 2, and so forth. To import the segmentations, you place the label images in a folder with the same name as the image sequence. Then you place the folder in a folder with a name start-

ing with *Segmentation*, and place that folder in the *Analysis* folder. As an example, the label images could be placed in a folder named *[experiment folder]/Analysis/Segmentation_CellProfiler/[sequence name]*. When you have done this, you can open the segmentation GUI and select the segmentation algorithm "Segment_import". There is also a different segmentation algorithm called "Segment_import_binary" which converts the label images into binary images. The advantage of this is that you can use post-processing algorithms from the segmentation GUI, and the disadvantage is that it merges adjacent regions. The segmentation algorithms "Segment_import" and "Segment_import_binary" are not shown unless you have a segmentation that can be imported.

In addition to importing a segmentation, you can also import features of segmented regions from CellProfiler. To do this, you need to export the features from CellProfiler to a file called *labels.csv*, and put the file in the same folder as the label images. Then the features will be imported together with the segmented regions when you run the tracking. The imported features can currently not be used to train classifiers through the program, but you can write your own MATLAB-scripts to train such classifiers.

10.1.4 Saving segmentation settings

To use the segmentation GUI successfully, it is important to understand how segmentation settings are saved to the settings files. The settings that you enter are not automatically saved to the settings files and when you switch to a new image sequence, the settings that you see are not necessarily the settings that have been saved for that file.

The settings that you enter are saved to the settings file first when you press one of the buttons, **Save seq**, **Save list** or **Save all**. **Save seq** applies the settings to the currently displayed image sequence, **Save list** lets you pick what image sequences the settings should be applied to, and **Save all** applies the settings to all image sequences.

When you switch to a new image sequence using either the **Previous**-button, the **Next**-button, or the image sequence dropdown menu, the settings in the GUI will not be changed unless the **Revert to saved**-button is down. Not having the button pressed down lets you verify that a new set of segmentation settings work well on all image sequences in a dataset, without saving it. You can always load the saved settings for the displayed image sequence by pressing the **Revert to saved**-button twice.

10.2 Segmentation algorithms

The user interface has many different segmentation algorithms that can be selected under "SegAlgorithm". When an algorithm is selected, settings specific to that algorithm will be displayed and settings specific to other algo-

rithms will be hidden. The names of settings specific to a particular segmentation algorithm start with one or multiple letters identifying the segmentation algorithm. In general, "Segment_localvariance" works well for transmission microscopy images, "Segment_bandpass"/"Segment_bandpass3D" works well for fluorescence microscopy images, and "Segment_fibers" works well for tissue sections with a staining on the cell membranes. The following sub-sections describe the most commonly used segmentation algorithms in detail.

10.2.1 Segment_threshold/Segment_threshold3D

Thresholding is a simple segmentation algorithm which assumes that the objects of interest are brighter (or darker) than the background. The algorithm creates regions by first labeling all pixels with an intensity above (or below) a threshold as foreground pixels and then computing the connected components of the foreground pixels. This algorithm can work well on fluorescence microscopy images, but usually requires Gaussian smoothing, which is described in Section 10.3.2. The algorithm "Segment_threshold" is used for 2D data and "Segment_threshold3D" is used for 3D data, but mathematically the two algorithms are equivalent.

10.2.2 Segment_localvariance

Local variance segmentation computes the sample variance of the pixel intensities in a small region around every pixel. The resulting local variance image is then thresholded to produce a segmentation. The algorithm relies on having a background with a relatively uniform intensity. The algorithm can handle transmission microscopy images created using bright field microscopy, phase contrast microscopy and differential interference contrast (DIC) microscopy. It can also handle fluorescence microscopy and other microscopy techniques where the cells are brighter than the background, but bandpass filtering is likely to work better for such images. Table 2 explains the settings associated with local variance segmentation. The algorithm can detect parts of cells with very low contrast, such as filopodia and lamellipodia but it has problems separating cells in clusters.

10.2.3 Segment_bandpass/Segment_bandpass3D

Bandpass filtering will usually work well for images where the cells are consistently brighter than the background. It works well for fluorescence microscopy but also for some transmission microscopy techniques such as dark field microscopy and oblique illumination microscopy. The algorithm applies a bandpass filter to the spatial frequencies of the images. This will reduce noise by removing the high frequencies and reduce non-uniform illumination by removing low frequencies. To perform the bandpass filtering, the

Table 2: Settings associated with local variance segmentation.

LVSegRegionSize	Radius of the region around each pixel, in which the sample variance is computed.
LVSegThreshold	Threshold applied to the sample variance. Before thresholding, the sample variance is rescaled using the function $f(x) = \log(x + 1)$ to give values in a range from 1 to about 5.
LVSegErodeSize	The number of pixels that will be removed around the borders of all regions using erosion.
LVSegRegionShape	The shape of the region in which the sample variance will be computed. The available options are "square", "round", and "gaussian". "square" is fastest and usually works well. "gaussian" gives the smoothest borders for large regions. The option "gaussian" puts larger weights on pixels close to the center of the region than pixels further away in the computation of the sample variance.
LVSegErodeShape	The shape of the structuring element used for erosion. The available options are "square" and "round".

algorithm first computes a smoothed image I_{LP_1} by convolving the original image with a Gaussian kernel with a small standard deviation. Then a background image I_{LP_2} is created by convolving the image with a Gaussian kernel with a larger standard deviation. The bandpass filtered image is then computed as

$$I_{BP} = I_{LP_1} - \alpha I_{LP_2}. \tag{1}$$

Here, α is a free parameter which adds more flexibility. Setting $\alpha = 1$ results in a true bandpass filter. The algorithm "Segment_bandpass" is used for 2D data and "Segment_bandpass3D" is used for 3D data, but mathematically the two algorithms are equivalent.

Compared to thresholding, bandpass filtering performs better on images with non-uniform illumination, large intensity variations between objects, and objects that form clusters. The only weakness of bandpass filtering that thresholding does not have is that parts of very dim objects that are close to bright objects can disappear from the segmentation. Table 3 explains the settings associated with bandpass segmentation.

Table 3: Settings associated with bandpass segmentation.

BPSegHighStd	The standard deviation of the Gaussian kernel used to create the background image I_{LP_2}. A larger value results in a more blurred I_{LP_2}. The value is in pixels and is usually set between 5 and 1000.
BPSegLowStd	The standard deviation of the Gaussian kernel used to remove noise. Increasing the value removes more noise but also blurs the image. The value is in pixels and is usually set between 1 and 5. The value has to be less than BPSegHighStd.
BPSegBgFactor	α in Equation (1).
BPSegThreshold	Threshold applied to the bandpass filtered image. The value is usually set between 0.1 and 0.0001.

10.2.4 Segment_fibers

Segmentation of muscle fibers is done by applying a watershed transform to a staining of the cell membranes and then merging regions where the intensity on the border between the regions is low compared to the average intensities inside the regions. Table 4 explains the three most important settings associated with this segmentation algorithm. There are several other settings, but they are usually less important for the performance. Make sure that "SegChannel" is set to the channel containing the membrane staining.

10.3 Preprocessing

Sometimes, the images need to be preprocessed before the segmentation algorithms can be applied. This usually means removing a static background using background subtraction or applying smoothing to reduce noise. The image stabilization described in Section 8 is also a type of preprocessing, but the stabilization cannot be previewed in this GUI. If you want to segment cells in microwells, imaged using transmission microscopy and a moving stage, you need to stabilize the image sequence before you can see what the segmentation will be like. You can still get an idea about the segmentation performance from the un-stabilized image sequence, but the background subtraction algorithms described in Section 10.3.1 work very poorly without stabilization.

Table 4: Settings associated with segmentation of fibers.

FibSegBgThreshold	Intensity threshold below which pixels are considered to belong to the background.
FibSegMergeThreshold	Intensity ratio threshold for merging of adjacent regions. Increasing this value will merge more regions.
FibSegShapeHMin	The segmented regions are broken into smaller regions using a watershed transform applied to the distance image of the segmentation mask. This parameter specifies the h-value of an h-min transform applied to the distance image before the watershed transform is computed. Higher values result in fewer region breaks.

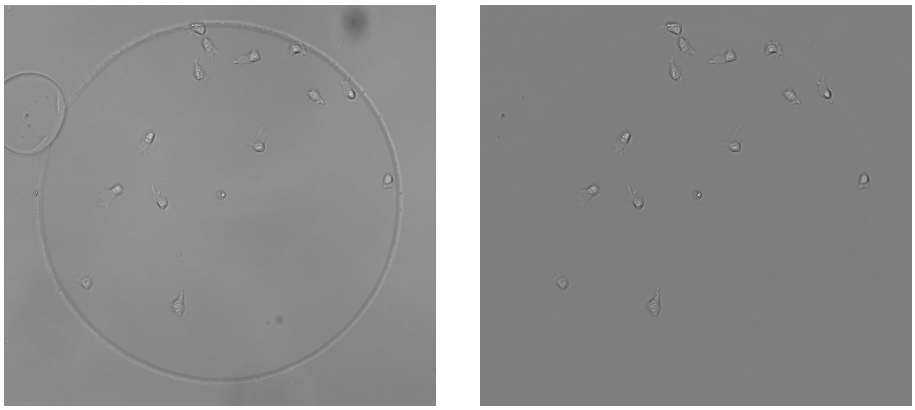
10.3.1 Background subtraction

If there are prominent background features, such as the outline of a microwell, it is often necessary to perform background subtraction before the image can be segmented. The most commonly used background subtraction algorithm is called "BgSub_median". This algorithm creates a background image by computing a median image over the time dimension of the image sequence, and then subtracts this background image from the individual images of the image sequence. Once the background has been subtracted, regions with a lot of background, such as the microwell, can be attenuated. This will remove artifacts caused by media changes or movements in hydrogel substrates. The amount of attenuation can be specified using the setting "BgSubAtten". Increasing "BgSubAtten" will remove more background residue, but it can also remove cells or parts of cells from the segmentation. The background subtraction can be visualized by selecting "bgSub" in the "Display" dropdown menu. Figure 7 shows an example where a circular microwell has been removed using background subtraction.

10.3.2 Smoothing

The image can be smoothed using a Gaussian kernel, prior to segmentation, by setting "SegSmooth" to a non-zero value, which specifies the standard deviation of the kernel. This is often necessary when threshold segmentation is used, but most other segmentation algorithms have built in mechanisms to handle noise.

Credit: *Penney Gilbert, Stanford University*



(a) (b)

Figure 7: MuSCs in a 600 μm microwell (a), and an image where the microwell has been removed using background subtraction (b).

10.4 Light correction

Setting "SegLightCorrect" to "additive" shifts the intensities of the images so that all images in a sequence get the same mean value. This can be useful if the lighting is not the same throughout the sequence, but otherwise it does not make a big difference.

10.5 Post-processing

After a segmentation algorithm has been applied, the segmentation can be refined using different post-processing methods. Table 5 describes the different settings that are available, and Section 10.5.1 explains how watershed transforms can be used to separate cells in clusters.

Table 5: Settings for segmentation post-processing.

SegFillHoles	Removes holes in the segmented regions.
SegMinHoleArea	Holes with fewer pixels are filled. The default value is inf, and that means that all holes are filled.
SegMinArea	Regions with fewer pixels will be removed.
SegMaxArea	Regions with more pixels will be removed.

SegMinSumIntensity	Regions with a summed image intensity below this threshold will be removed. This setting will often be more useful than SegMinArea for fluorescence images. The pixel values are normalized to be between 0 and 1 and the minimum pixel value in the image is subtracted from all pixels to get rid of background illumination.
SegClipping	Threshold for intensity clipping, between 0 and 1. First the image is normalized so that the highest possible value is 1. Then values above the threshold are replaced by the threshold, and finally the intensities are rescaled to the original range.
SegWatershed	Image parameter used in a seeded watershed transform for separation of cells in clusters.
SegWSmooth	Standard deviation of a Gaussian smoothing kernel applied before the watershed transform. Larger values create fewer fragments and smoother separation boundaries.
SegWHMax	Removes local watershed minima with a depth, relative to the surrounding, below this value. Larger values create fewer fragments.
SegWThresh	Removes local watershed minima with an absolute depth below this value. Larger values create fewer fragments, and "-inf" disables the setting.
SegWatershed2	Image parameter used in a second seeded watershed transform which separates cells in clusters further. The cell outlines generated by the first watershed transform are used as input. Parameters for this transform are specified by "SegWSmooth2", "SegWHMax2", and "SegWThresh2".

10.5.1 Watersheds

Most segmentation algorithms have problems separating cells in clusters into separate regions. One way of overcoming this problem is to apply a watershed transform to some property of the image which has low values in the middle of cells and high values at the cell borders and in the background.

The watershed transform will treat the image as a landscape and simulate flooding of the landscape with water. As the water level rises, lakes will form and the surfaces of the lakes correspond to separated cell regions.

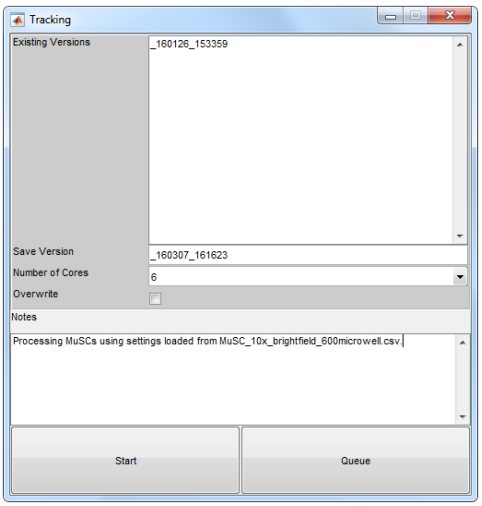


Figure 8: GUI for automated optimization of segmentation parameters.

10.6 Automated segmentation optimization

It is usually hard to find optimal segmentation parameters through manual tweaking. Therefore the BA has a GUI for automated optimization of segmentation parameters. To perform the optimization, you need to export a segmentation ground truth as described in Section 11.5.1. Then you click on **Settings/Optimize segmentation parameters**, to open the GUI shown in Figure 8. The options in the GUI are explained in Table 6.

Automated segmentation optimization is available for the segmentation algorithms Segment_localvariance, Segment_bandpass, Segment_bandpass3D, Segment_threshold, and Segment_threshold3D. The optimization algorithm optimizes either the SEG measure or the average of the SEG measure and the TRA measure. Both measures are described in Section 15. The optimization procedure starts from the parameters specified by the user and uses coordinate ascent to find better parameters. Each time better parameters are found, the parameters are saved to the settings files of all image sequences. The initial step length is 25 % of the start value for each parameter. When a parameter is changed, the step length is increased by 20 %. When a parameter cannot be changed to produce a better segmentation, the step length is decreased by 20 %. The parameters and the corresponding performance values are plotted in a separate window during the optimization. More information about the optimization procedure can be found in [1].

Table 6: Options for segmentation optimization.

Image sequences	Which image sequences to include in the optimization.
Parameters	Which segmentation parameters to optimize. The segmentation algorithm is selected in the GUI described in Section 10.1.
Scoring function	The scoring function to optimize.
Optimize individually	If this checkbox is checked, the segmentation parameters are optimized individually for all image sequences. Otherwise, they are optimized jointly.
Number of iterations	The number of times the optimization algorithm will cycle through all segmentation parameters.
Maximum number of images	The optimization will include at most this many images from each image sequence in the optimization. This setting can be used to reduce the run time of the optimization. The algorithm will use all images with a segmentation ground truth if the parameter is set to NaN.
Images with most cells	This setting is shown only if a maximum number of images is set. If the checkbox is checked, the optimization will use the images with most ground truth regions. Otherwise it will use randomly selected images.
Optimization path	Folder where intermediate results are saved.

11 Tracking

Pressing **Automated/Track** opens the GUI shown in Figure 9, where you can perform tracking. The segmentation GUI described in Section 10.1 only changes the segmentation settings and does not generate any segmentation

results. The tracking GUI will first run segmentation on all images in a sequence, and then link the segmented regions into tracks. Segmentation of tissue sections is treated as tracking of tissue regions in an image sequence consisting of a single image.

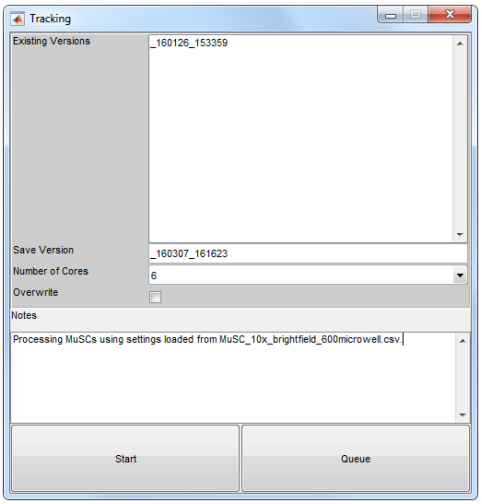


Figure 9: GUI which performs tracking.

In the **Save Version** text box, you can specify under what name the tracking results should be saved. The default name is the time point when the GUI was opened, in the format _YYMMDD_hhmmss, where YY = year, MM = month, DD = day, hh = hour in 24 hour format, mm = minute, and ss = second. The name can be changed to any combination of letters, digits, and underscores, but it can be good to keep the time point as a prefix so that the alphabetical order and the chronological order are the same. The **Notes** text box lets you write a short description of the tracking version, where you can describe what settings you used and why. This information is saved in the log file described in Section 11.5. The list box **Existing Versions** has a list of all tracking results that have been created previously. If you click on one of them, the corresponding version name and notes will be copied to the appropriate text boxes. That is useful when you want to resume an old processing session.

As for stabilization and cutting, there is a dropdown menu where you can specify how many processor cores should be used for parallel processing. If a single image sequence is tracked, the parallelization is done over time points in the segmentation step. Otherwise, a number of image sequences are processed in parallel, which means that all processing steps are parallelized.

Once the processing of an image sequence has finished, that image sequence will not be processed again if the processing is resumed, even if the settings have been changed. To force the program to redo old computations,

you can either specify a new name in **Save Version**, or check **Overwrite**. Resuming a processing session is described in Section 11.3.

11.1 Queue

Normally, the processing is started by pressing the **Start** button, but you can also put the computations in a processing queue by pressing **Queue**. This makes it possible to run multiple processing sessions in sequence without having to start all of them separately. This can be useful for example if processing is run overnight or over a weekend.

11.2 Printouts

The different processing functions will print progress messages to the command line. When parallel processing is used, progress messages from the different processing threads will be mixed. See section 6.1 for instructions on how to get printouts in the deployed application on Mac.

11.3 Stopping and restarting

You can stop the processing in MATLAB by pressing **Ctrl+c**. If the program is busy doing something in a mex-file, it can however take a long time before the execution actually stops. You can resume the processing at a later time by entering the old label in the **Save Version** text box. When the processing is resumed, already processed image sequences will not be processed again and the segmentation of partially processed image sequences will be reused. Make sure that **Overwrite** is not checked when you want to resume the processing.

11.4 Track linking settings

There are a number of settings associated with the track linking. The settings can be changed under **Settings/Settings** and the most important ones are described in Table 7.

Table 7: Settings associated with track linking.

countClassifier	Classifier used to estimate the number of cells in segmented regions.
splitClassifier	Classifier used to estimate the probability of mitosis in segmented regions.
deathClassifier	Classifier used to estimate the probability of apoptosis in segmented regions.

pCnt0	Fixed probability that segmented regions contain 0 cells, used only if countClassifier is set to "none".
pCnt1	Fixed probability that segmented regions contain 1 cell, used only if countClassifier is set to "none".
pCnt2	Fixed probability that segmented regions contain 2 or more cells, used only if countClassifier is set to "none".
pCntExtrap	Factor by which the probability is decreased when a cell is added to a region containing 2 or more cells.
pSplit	Fixed probability of mitosis in segmented regions, used only if splitClassifier is set to "none".
pDeath	Fixed probability of apoptosis in segmented regions, used only if deathClassifier is set to "none".
TrackPAppear	Fixed probability for a cell region to contain a cell that randomly appeared in the current time step. Cells can appear randomly by going out of suspension in the beginning of the experiment or by detaching from the substrate and getting washed into the field of view later in the experiment.
TrackPDisappear	Fixed probability for a cell region to contain a cell which will disappear randomly before the next image.
TrackMigInOut	If this is set to 1, cells are allowed to enter and leave the field of view through migration.
TrackXSpeedStd	Standard deviation, in pixels/voxels per frame, for the cell displacement between two images, in the x- and y-directions.
TrackZSpeedStd	Standard deviation, in voxel widths per frame, for the cell displacement between two images, in the z-direction of a 3D image sequence. The setting is specified in voxel widths, so if the motion is isotropic, the value should be equal to TrackXSpeedStd, even if the voxel height is different from the voxel width.

11.5 Tracking results

All of the tracking results are saved in a folder with the path *[Experiment folder]/Analysis/CellData[Save Version]*. The information about the tracks and the outlines of the cells is stored in a mat-file with the same name as the image sequence. The settings that were used are stored in a file named *Settings.csv*. Log files with information about the processing that was done are stored in a folder named *Logs*. The log files are txt-files named after the image sequences. The files contain notes written by the user, and information about the BA version and the MATLAB version. The folder *Compact* contains mat-files with tracking results where cell outlines have been left out. These files are usually much smaller than the files with the complete results. The folder *Resume* contains intermediate processing results, primarily segmentation results, that can be used to resume an interrupted processing session. The folder *Tracking_log* contains track linking information which is used only for development and debugging.

For large datasets, such as 3D datasets with hundreds of frames and thousands of cells, the saving of the mat-file can fail due to an out-of-memory error during the compression step. If that happens, the results are instead saved to tif-files using the format described in Section 11.5.1.

11.5.1 Exporting tracking results in the CTC format

A file format based on tif-images was used in the ISBI Cell Tracking Challenges, which were organized in connection with the International Symposium on Biomedical Imaging in 2013, 2014, and 2015. In the format, label images are saved to 16-bit tif-images, where the background pixels are 0, the pixels of object one are 1, the pixels of object two are 2, and so forth. For 3D datasets, a tif-stack with one slice for each z-slice is saved. Additional tracking information is saved in a space separated txt-file with 4 columns. The first column is the track index, the second column is the start frame, the third column is the end frame, and the fourth column is the track index of the parent cell, or 0 if the track does not have a parent. Frame indices are 0-based and track indices are 1-based. The format is described in detail in [5–7].

We have extended the format to incorporate information about death events and false positive regions which are created in the segmentation step but not included in a track in the tracking step. Information about death events are stored in a file named *deaths.txt*. This file has two columns. The first column is the track index and the second column is 1 if the track ended in death and 0 otherwise. False positives are saved in a subfolder named *false_positives*. In that folder, information about false positive tracks are saved in the same format as the real cell objects.

The CTC format can be used so save both tracking results, tracking

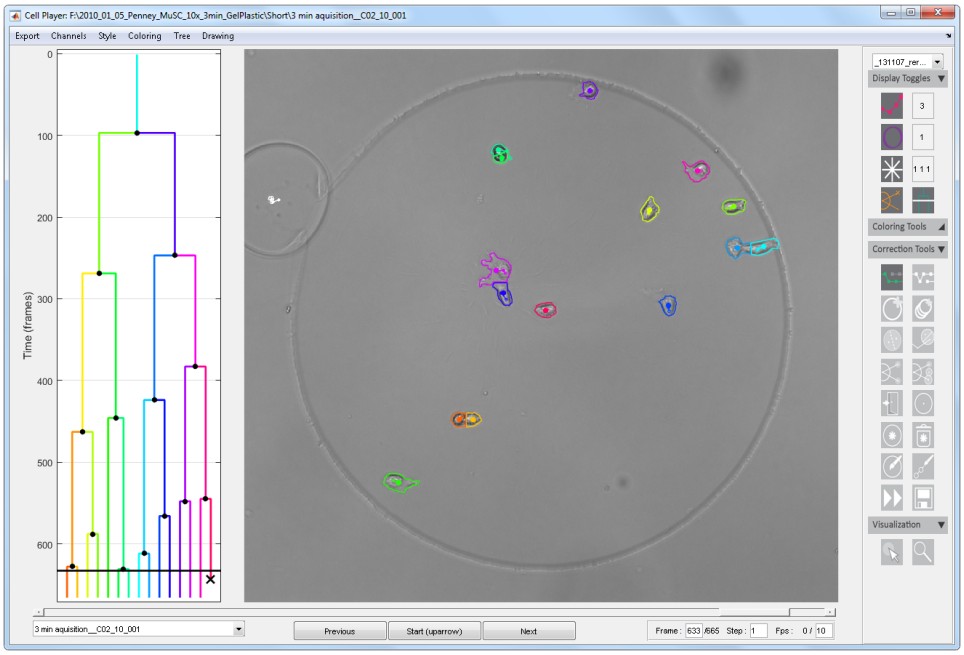
ground truths, and segmentation ground truths. Tracking results and tracking ground truths are saved in the same format, except that the files are named differently. In the segmentation ground truths, 16-bit tif images are used to represent the segmented regions, but there is no track information, and the region numbering goes from 1 to the number of regions in each image. Another difference is that 3D segmentation ground truths can be defined in a single z-plane, but this feature is not yet supported by the BA.

The export to the CTC format is performed by selecting the image sequences that the export should be performed for in the main GUI, and then clicking **Export tracks to CTC format/RES tracks**, **Export tracks to CTC format/SEG ground truth**, or **Export tracks to CTC format/TRA ground truth** to export tracking results, segmentation ground truths, or tracking ground truths respectively. Then the user must select an existing tracking result to export. Tracking results are exported to a subfolder named *RES/[Image Sequence]_RES* in the CellData-folder where mat-files with tracking results are normally saved. Ground truths are exported to a folder named *[Image sequence]_GT* in the Analysis folder of the experiment. The tracking ground truth is saved in a subfolder named *TRA* and the segmentation ground truth is saved in a subfolder named *SEG*. Tracking results can be saved to the CTC format automatically after the tracking step has been executed by setting the option "TrackSaveCTC" to 1 on the advanced level of the tracking settings.

12 Manual Correction

Once the cells have been tracked, the tracks and the outlines of the cells can be visualized and modified in a manual correction interface found under **Manual/Track Correction** and shown in Figure 10. When you open the GUI, no tracking version is selected, so you have to select the desired tracking version in the dropdown menu at the top of the control panel. The outlines of the cells can be visualized independently of the tracks, and the lineage tree can be shown next to the played image sequence. The cells are color coded, so that a cell in the lineage tree can be identified in the image sequence, when the number of cells is small. False positive detections that were found by the segmentation algorithm, but not included in any cell track by the track linking algorithm are shown in white, or a different color specified by the user.

The **Style** menu specifies how tracks and outlines should be plotted. The **Default** style is normally used for correction, as is marks cell centroids in the previous and the current image, and as it uses thin lines which do not cover the cells. The **Save** style has thicker lines and visualizes all cell centroids as filled circles. This style is meant to be used when image sequences are recorded. The **ISBI** style uses thin unfilled circles to visualize cell centroids



Credit: *Penney Gilbert, Stanford University*

Figure 10: GUI for visualization and correction of cell tracks and cell outlines.

and is therefore suitable for images with small objects that would be covered by filled circles.

The **Coloring** menu has three different coloring alternatives for the cells. The **Rainbow** coloring applies colors from a rainbow to the horizontal axis of the lineage tree. The **Random Hues** coloring gives the cells random colors taken from a rainbow. This alternative is often good during correction, as nearby cells are usually given different colors, and as the colors do not change when the lineage tree is changed due to correction. The **Random Colors** coloring picks random colors from a set of 6 colors.

The controls on the control panel to the right of the image have been grouped into the four groups **Display Toggles**, **Coloring Tools**, **Correction Tools**, and **Visualization**. The groups are described in Sections 12.1, 12.2, 12.3, and 12.4 respectively.

It is important to think about what you want to use the tracking results for. For analysis of cell counts and population averages of parameters like cell size and cell speed, it is usually not necessary to perform correction at all, but for some analysis of individual cells and for analysis of lineage tree parameters, it can be very important to perform correction.

12.1 Display Toggles

The buttons in this group are all toggle buttons which toggle plotting of different types of data on and off. There are also text boxes to specify for how many images back in time tracks and outlines should be plotted. Furthermore, there is a text box where the color of false positive detections can be specified as an RGB triplet.

12.2 Coloring Tools

The coloring tools let you specify the colors of individual cells. The **Change Color of Cell** tool lets you specify a color as an RGB triplet and then click on cells that should have that color. The **Change Color of All Cells** tool changes the color of all cells at once.

12.3 Correction Tools

This group contains tools for editing of cell tracks and outlines. Some of the tools perform simple track operations and others speed up the correction of specific tracking errors. Table 8 explains the different tools. The references to large and small circles assumes that the "default" display setting is used. Large filled circles are detections in the current image, small filled circles are detections in the previous image, and small unfilled circles are detections in even earlier images. The segmentation editing tool is described in detail in Section 12.3.1.

Table 8: Controls used for correction of cell tracks.

Track Tool	Allows you to change the track links drawn between detections. To change a link in a track, first click on the small filled circle in one end of the line and then click on the large filled circle that you want to connect it to.
Single Frame Track Tool	Allows you to move a single detection in a cell track. You can move it to another blob or you can place it in the background where there are no blobs, but then a new point blob without an outline will be created. Moving a cell inside the same blob has no effect.
Add Point	This tool adds a new cell with a point blob to a single image.

Continuously Add Points	This tool lets you create a new track by clicking where the cell should be in each frame. When you have clicked in one frame, the player will go to the next frame in the sequence automatically, so that you can generate an entire track through a sequence of clicks. If you click on a cell region, the cell is added to that region. If you click in the background, a point blob is created where you clicked. To extend an existing track, you can click on a small filled dot with the Track Tool selected and then switch to the Continuously Add Points tool.
Split	Copies an entire cell track. This will create a new cell that appears in the same blob as the copied cell in all frames where the copied cell is present. The blob of the original cell will then be split between the two cells.
Track Split	This tool lets you extend an existing track by copying a part of another track. You first click on the small filled circle of the cell that you want to extend and then on the large filled circle of the track that you want to copy a part of. The tool will copy the second cell that you click on from the current frame to the end of the track in the same way as the **Split** tool. The copied part of the track is then connected to the end of the first cell that you clicked on.
Set Children	Allows you to specify the parent-child relationships between cells. First click on the small filled circle of the parent cell and then click on the large filled circles of the two child cells. You have to specify two children.
Set Split Children	This tool lets you specify parent-child relationships and at the same time create the child cells by copying parts of other tracks. The copy mechanism works in the same way as it does in the **Track Split** tool.

Mark Leaving Cell	This tool toggles the fate of a cell between dying and leaving the field of view. You click on the large filled circle of the cell in any image.
Move Mitosis	Moves a mitotic event associated with a clicked cell to the current image. If there are multiple mitotic events, the one closest in time to the current image will be moved. If the clicked cell is the parent cell in the closest mitotic event, the two child cells will both follow the track of the parent cell between the time points of the new and the old miotic events. If the clicked cell is one of the child cells in the mitotic event, the parent cell will follow the track of the clicked child cell between the time points of the new and the old mitotic events. The track of the other child cell will be turned into a false positive track between the same time points.
Delete	Right-clicking on a cell will turn the whole cell into a false positive cell. This will also break the connection to the parent of the cell, and turn all children of the cell into false positive cells. Left-clicking on a cell will instead kill the cell in the current frame and keep all prior time points. This will also turn all children into false positive cells. Right-clicking or left-clicking a false positive track will turn the entire track or all subsequent frames into a real cell. All children will also be turned into real cells.
Edit/Draw Segments	Turns editing of blob segments on, so that the blobs can be edited as in a drawing program. This tool is explained further in Section 12.3.1
Remove FP	Pressing this button will remove all false positive blobs which do not have outlines associated with them. False positive track fragments with outlines will be kept.

Jump	Moves to the next image where the number of cells changes.
Save	Opens a dialog box for saving of the corrected tracking result. This tool is explained further in Section 12.3.2.

12.3.1 Correcting outlines

When the **Edit/Draw Segments**-button is pressed down, you can modify the outlines of the cells in the current image. The mouse acts as a paint brush when the left mouse button is pressed down and as an eraser when the right button is pressed down. The size of the brush and the eraser can be changed using the scroll wheel on the mouse or using the + and - keys on the keyboard.

The **Drawing** menu has several options that affect how outlines are edited. The **Drawing/Create** menu specifies if new regions that are created should become real cells or false positives. The **Drawing/Holes** menu specifies if holes in regions that are drawn should be filled or not. The **Drawing/Merging** menu has three alternatives which specify what should happen if you draw on multiple regions at the same time. With the **Re-break** alternative, the regions that are merged by the drawing will be split between the cells using the k-means clustering algorithm used to split regions during the tracking phase. With the **Combine** alternative, the outline of the largest cell is replaced by the union of the drawn region and all blobs that were drawn on. The smaller cells are turned into false positives, and their outlines are removed. With the **Overwrite** alternative, the blob that you started drawing on will be expanded to the drawn region, and take over pixels from other blobs if the pixels are already taken.

12.3.2 Saving corrected data

When either the tracks or the outlines have been edited, the **Save**-button can be pressed to save the new cell data. Pressing the **Save**-button opens a dialog box where you can specify a name for the new tracking. It is good to add a tag, such as "Corr", to the corrected tracking versions, to keep track of which versions have been corrected. It is also good to keep the original tracking version, in case it is needed for performance evaluation or some other purpose. Files are overwritten without notifications.

12.4 Visualization

The **Visualization** tool group contains a cell selection tool and a zoom tool. The cell selection tool colors the selected cell orange and all other cells blue. This makes it possible to see where a specific cell has been at earlier time points, and to identify a cell in the lineage tree with a cell in the images. The zoom button enables the zoom tools and the 3D slice selection tools which are always enabled in other players.

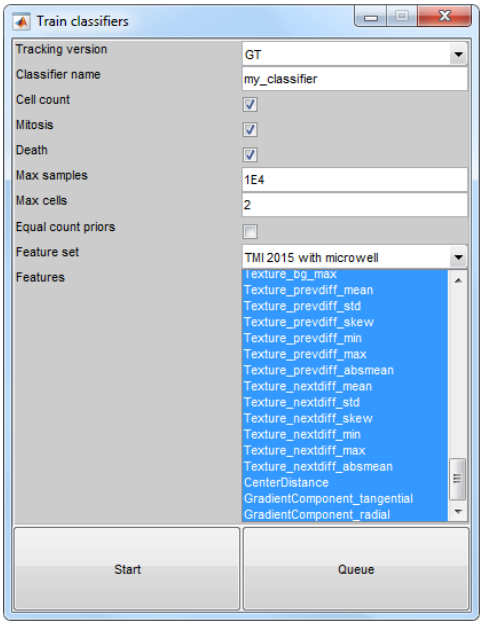


Figure 11: GUI for training of classifiers.

13 Training classifiers

The track linking algorithm can use classifiers to estimate how likely different events are to occur in the regions created by the segmentation algorithm. The program has classifiers that estimate the probability of mitosis, apoptosis and different cell counts inside the individual regions. The classifiers have been created specifically for a particular cell type and a particular imaging technique. If the data that is about to be processed is too different from the data that was used to create the classifier, the tracking performance will be poor. If there is no classifier that matches the data to be processed, it is usually best to set "countClassifier", "splitClassifier", and "deathClassifier" in Table 7 to "none" and instead specify the settings "pCnt0", "pCnt1", "pCnt2", "pSplit", and "pDeath" appropriately. Once you have one or

more image sequences with corrected cell tracks, you can however create your own classifiers by clicking **Settings/Train classifiers**. This opens the GUI shown in Figure 11. The different options in the GUI are explained in Table 9. The trained classifiers are based on multinomial logistic regression. Training of classifiers can require large amounts of corrected data, especially when classifiers for mitosis and apoptosis are trained.

Table 9: Options for training of classifiers.

Tracking version	Corrected tracking version to use for training.
Classifier name	Name of the new classifier. The count, mitosis, and death classifiers are given the same name.
Cell count	Trains a cell count classifier.
Mitosis	Trains a mitosis classifier.
Death	Trains a cell death classifier.
Max samples	The maximum number of training examples in each class. Using more training examples usually gives a better classifier, but the training takes longer. There is usually no point in using more than 1E5 training examples. The training examples are drawn randomly if there are more examples than the specified number. In that case, the training examples are weighted in the training, so that the original balance between the classes is maintained.
Max cells	The maximum number of cells that the cell count classifier can handle in a region. If the setting is 2, the classes of the classifier will be 0 cells, 1 cell, and 2 or more cells. A maximum of 2 is usually good, as there are often too few examples of regions with 3 cells to train a reliable classifier.
Equal count priors	If this checkbox is checked, the priors of the classes in the cell count classifier will be set equal. This can be good in inhomogeneous datasets where different parts have very different priors.
Feature set	The set of region features to use for classification.
Features	The features in the selected feature set.

13.1 Features

The region features are grouped into feature sets that can be selected in the **Feature set** dropdown menu. The feature set "ISBI 2012" is taken from [3] and the feature set "TMI 2015" is taken from [4]. The feature set "All features" is the union of all features in "ISBI 2012" and "TMI 2015". All of these feature sets were designed for tracking of adherent cells in bright field microscopy, but they should work well for other cell types and imaging techniques as well. All of the feature sets have options in the dropdown menu with and without microwells. The only difference is that the feature "CenterDistance" has been added to the options with microwells. These options should only be used if you are tracking cells in circular microwells. You can select a subset of the features in a feature set by marking the features you want in the **Feature** listbox.

Table 10: Settings associated with the analysis of cell tracks.

condition	The name of the experimental condition for each image sequence. This affects how the image sequences are grouped in the analysis.
pixelSize	Pixel size of the camera sensor in μm. The pixels are assumed to be square.
magnification	The magnification of the camera objective.
dT	The time in seconds between image sequence frames.
startT	Time in hours when the imaging was started. The zero time point is usually defined as the time when the cells were plated.
minPlotT	The first time point (in hours) that should be included in analysis plots. If this is left empty, the plots start with the first image.
maxPlotT	The last time point (in hours) that should be included in analysis plots. If this is left empty, the plots end with the last image.

14 Analysis

Once the tracks and outlines of all cells have been computed and possibly corrected, different types of analysis can be performed on the data. The pro-

gram already has a lot of analysis functions for commonly used readouts, but for more complex and application specific readouts, custom analysis functions can be developed and easily incorporated into the program. Settings associated with analysis are explained in Table 10. To get the right conversions from pixels and frames to micrometers and hours, you need to specify the settings "pixelSize", "magnification" and "dT". Otherwise all results will be given in pixels and frames, even if the axis labels and printouts say "micrometers" and "hours".

14.1 Cell analysis GUI

Under **Analysis/Cell Analysis GUI**, you find the GUI shown in Figure 12, which is used for plotting of parameters of individual cells over time. The plots include data from a particular clone or image sequence and have a curve for each cell. Examples of parameters that can be plotted are axis ratios, lineage trees, cell sizes and the total distance traveled by each cell. The **Tracking Version** dropdown menu specifies what tracking version should be used for the analysis. Pressing **Start** cycles through the image sequences or the clones and shows a plot for each of them.

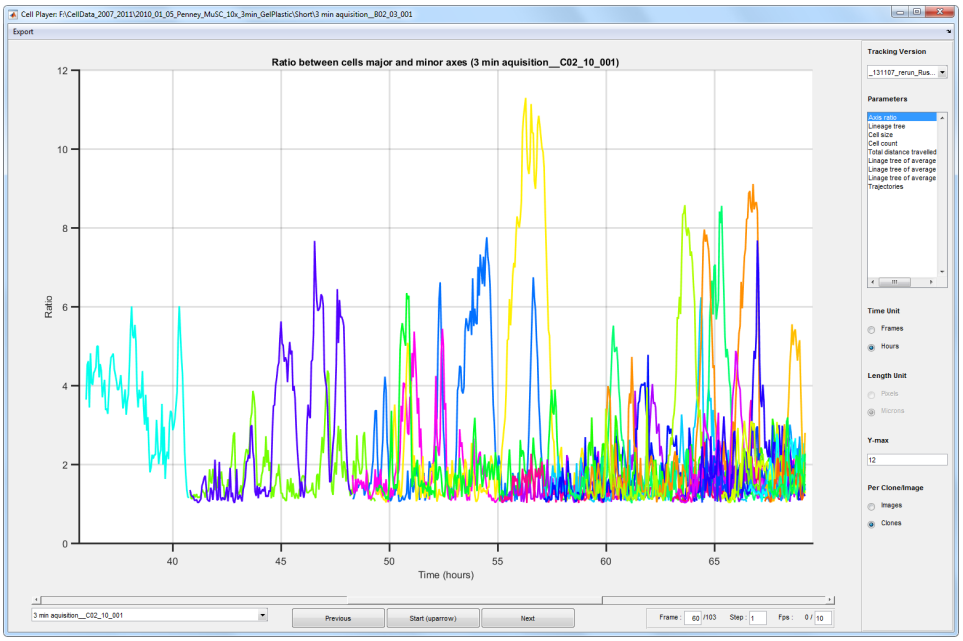


Figure 12: GUI for plotting of parameters of individual cells over time. The displayed plot shows axis ratios of muscle stem cells as a function of time.

14.2 Population analysis GUI

Under **Analysis/Population Analysis GUI**, you find the GUI shown in Figure 13, which is used for plotting of parameters averaged over the life spans of the cells. Every cell in the dataset will therefore have a value associated with it and the distributions of the values can be compared between different experimental conditions. There are a number of parameters, such as average speed, average axis ratio, and average size, that can be visualized. There are also a number of different plotting methods that can be used for visualization.

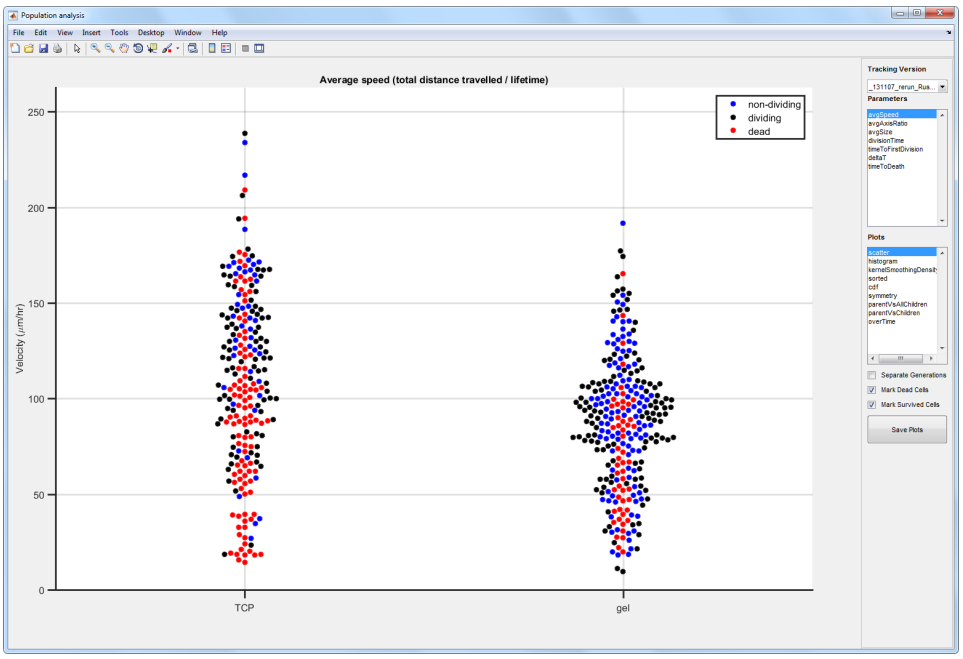


Figure 13: GUI for plotting of parameters averaged over the life spans of cells. The displayed plot shows the average speeds of muscle stem cells.

14.3 Scatter Plot Analysis GUI

While the Population Analysis GUI plots one cell parameter at a time, the Scatter Plot Analysis GUI lets you plot pairs of cell parameters against each other, to see how different parameters are connected. Figure 14 shows the GUI. You can choose what tracking version to analyze, which parameters to plot on the two axes, and how the cells should be colored. If you choose to color the cells by fate, dividing cells are black, non-dividing cells are blue, and dying cells are red. By unchecking the checkboxes "Mark Dead Cells" and "Mark Survived Cells", you can however color the dying and surviving

cells black like the dividing cells.

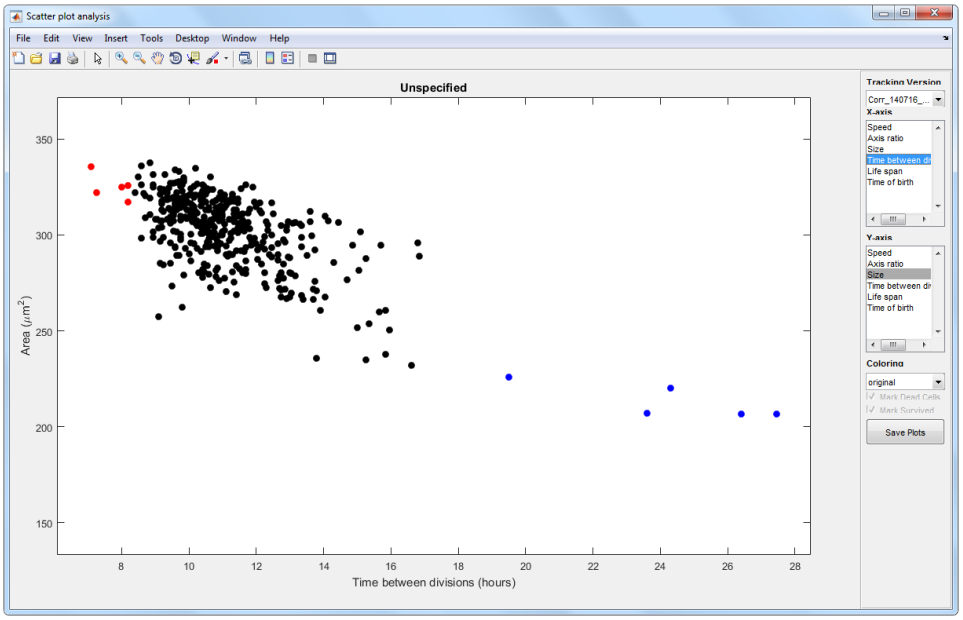


Figure 14: GUI where you can plot pairs of cell parameter against each other. The plot shows cell size against the time between divisions for a clone of hematopoietic stem cells. The cells were colored in the manual correction GUI.

14.4 Plot GUI

Under **Analysis/Plot GUI**, you find the GUI shown in Figure 15, were you can select analysis functions from a list and apply them to all image sequences or a subset of them. While the Cell Analysis GUI and the Population Analysis GUI plot raw parameter values for individual cells, most of the analysis functions in Plot GUI compile the parameter values into plots which quantify different cell behaviours. The tracking results are loaded only once even if multiple functions are executed, and all of the functions operate on the compact tracking results without blob outlines, which load faster than the full tracking results. Therefore, the loading of tracking results is relatively fast. The different parts of the GUI are explained in Table 11.

14.5 Exporting statistics to csv-files

If you want to perform statistical analysis on the tracking results, without writing your own MATLAB code, you can export data for different cell

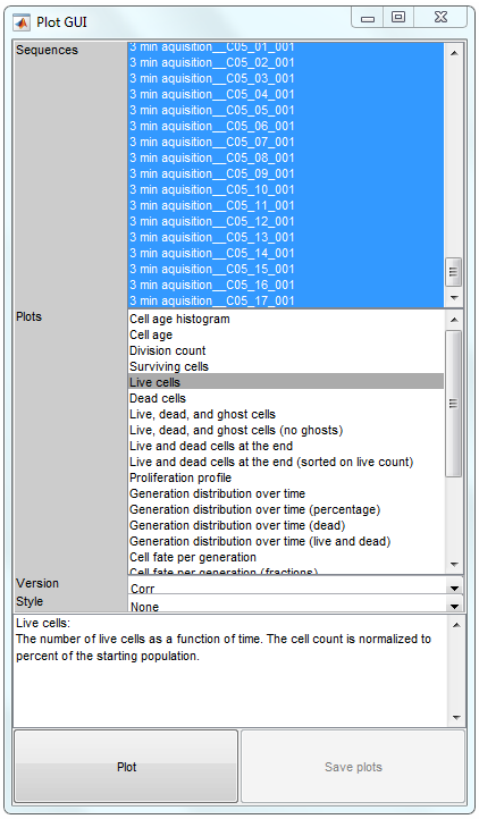


Figure 15: GUI where you can apply analysis functions to cells in tracked image sequences.

parameters to csv-files by pressing **Analysis/Statistics**. That will export the mean, standard deviation, median, minimum, and maximum of the cell parameters average speed, average axis ratio, average size, time between divisions, and time to first division. The csv-files will contain tables where the cells are grouped by experimental condition, and other tables where the cells are grouped both by experimental condition and cell generation. The number of cells in each group is also stated, so that you can perform statistical tests. The csv-files can be opened in Excel and in text editors.

15 Performance evaluation

In the BA, segmentation performance and tracking performance can be evaluated using the measures SEG and TRA, which were used to score algorithm performance in the ISBI Cell Tracking Challenges. The BA has its own MATLAB implementations of the two measures.

Before the performance evaluation can be performed, a segmentation or

Table 11: Controls in the Plot GUI.

Sequences	List where a set of image sequences can be selected for analysis.
Plots	List where one or more potting functions can be selected for execution. When a plotting function has been selected, a description of the function appears in the text box above the buttons at the bottom of the window.
Version	The name of the tracking version that will be loaded.
Style	Plotting style that will be applied to the plots after the analysis functions have been executed. The style "Screen" has thin lines, small markers, and a grid, for detailed analysis on a computer screen. The style "Print" has thicker lines, larger makers, and larger fonts, for legible printing on a full page. The style "Publication" has even thicker lines, larger markers, and larger fonts, for legible figures in a publication. If you do not select a style, you will get whatever comes out of the plotting functions, and that is usually similar to the style "Print".
Plot	Executes all the selected plotting functions on the selected image sequences.
Save Plots	Once you have run the plotting functions, you can press this button to open a GUI where you can save the plots to files.

tracking ground truth has to be created and exported in the CTC format, as described in Section 11.5.1. The tracking results to be evaluated are exported automatically to the CTC format if that has not been done already. The performance information is saved to log files in the *RES* folders described in Section 11.5.1, so that it does not need to be recomputed every time the user wants to review the performance. It is however necessary to delete the log files manually if the ground truth files are changed.

15.1 SEG measure

The segmentation performance can be evaluated using the SEG measure by clicking **Analysis/SEG Segmentation performance** with advanced menus turned on. The SEG measure computes the average Jaccard index between ground truth regions and matching segmented regions. A segmented region is considered matching if it covers more than 50 % of the ground truth

Table 12: Penalties for tracking errors in the AOGM measure.

Error	Penalty
False negative	10
Merged regions	5
False positives	1
Missing links	1.5
Incorrect links	1
Links of the wrong type	1

region. The Jaccard index is the number of pixels in the intersection of the regions divided by the number of pixels in the union of the regions. Ground truth regions with no matching segmented region are given a Jaccard index of 0. An advantage of the performance measure is it is enough if the ground truth contains a representative subset of cells. A disadvantage is that it does not penalize false positives. The measure is described in detail in [5–7].

The SEG measure disfavors segmentation algorithms with severe under-segmentation, as segmented regions are disregarded if they cover 50 % or less of the ground truth region. To overcome this problem, we provide a relaxed SEG measure which matches each ground truth region to the segmented region which gives the highest possible Jaccard index, without requiring an overlap of more than 50 %. The relaxed SEG measure is used if the check box named **Relaxed SEG measure** is checked. Otherwise, the traditional SEG measure is used.

15.2 TRA measure

The track linking performance can be evaluated using the TRA measure by clicking **Analysis/TRA Tracking performance** with advanced menus turned on. The TRA measure is a normalized version of the AOGM measure [9]. The AOGM measure is a weighted sum of tracking errors where the weights are taken from Table 12. The weights are roughly proportional to the time it takes to manually correct errors of the different types. The measure as a whole is therefore a measure of how long it takes to correct the tracking results manually.

The TRA measure is computed by normalizing the AOGM measure according to

$$\mathrm{TRA} = 1 - \frac{\min(\mathrm{AOGM}, \mathrm{AOGM}_0)}{\mathrm{AOGM}_0}, \tag{2}$$

where AOGM_0 is the AOGM measure of an empty tracking result. The TRA measure is described in [6, 7].

16 Specific types of data

This section describes methods and settings that can be used for specific types of data that have been processed using the BA in the past. Section 16.1 is about tracking of muscle stem cells (MuSCs) imaged using bright field microscopy, Section 16.2 is about analysis of fluorescently stained histological sections of muscle fibers, and Section 16.3 is about analysis of myotube fusion assays with fluorescently stained images.

16.1 MuSCs

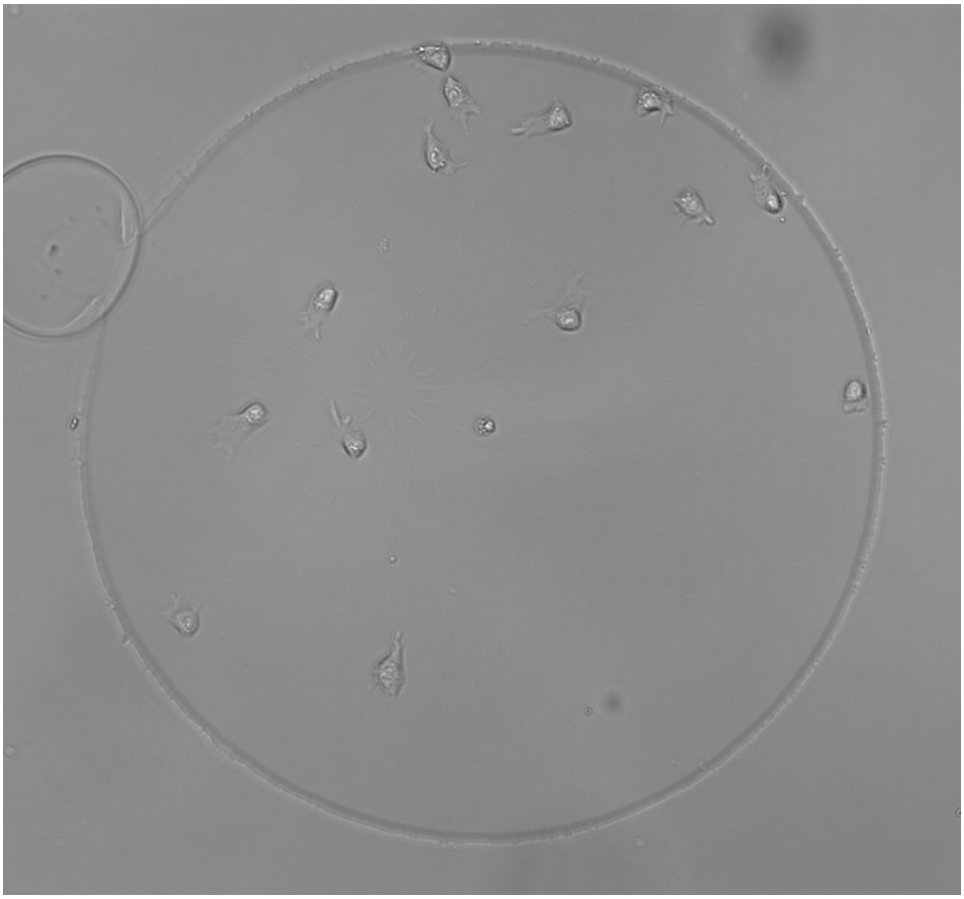
This section describes how to process image sequences of MuSCs, imaged using bright field microscopy. It is assumed that the cells are grown in circular microwells with a radius of approximately 600 μm and that they are imaged every 3 minutes using 10$\times$ magnification. Figure 16 shows an image from such an image sequence.

To process data of the type described above, you can go to **Settings/ Load Settings** and load the file *MuSC_10x_brightfield_600microwell.csv*, which has settings optimized for this type of data. To process data with an imaging interval of 5 minutes, you should increase "TrackXSpeedStd" in Table 7 and "dT" in Table 10 by a factor of 5/3. If you want to track cells grown on a flat culture substrate without microwells, you need to change "minWellR" and "maxWellR" in Table 1 to "nan". The settings for tracking of MuSCs will work well for other cell types imaged using 10$\times$ bright field microscopy as well, as long as the cells look and behave similarly to MuSCs. We have for example tracked myoblasts with good results.

The program will use background subtraction to remove the microwell outline during the segmentation step and therefore it is crucial that you apply image stabilization, as described in Section 8. Otherwise the segmentation will fail completely. If you changed the culturing media during the experiment, the background subtraction 10.3.1 can be problematic. If you find that the segmentation algorithm picks up the microwell boundary, you can try increasing the "BgSubAtten" setting associated with the background subtraction, or increase the "LVSegThreshold" setting associated with the local variance segmentation algorithm. The local variance segmentation algorithm is robust to differences between image sequences, so it should be possible to use the same segmentation settings for all image sequences in an entire experiment.

16.2 Muscle fibers

This section describes how to process fluorescently stained histological sections of muscle fibers. To segment the fibers, the program requires a staining of laminin or some other protein marking the outlines of the fibers. It does

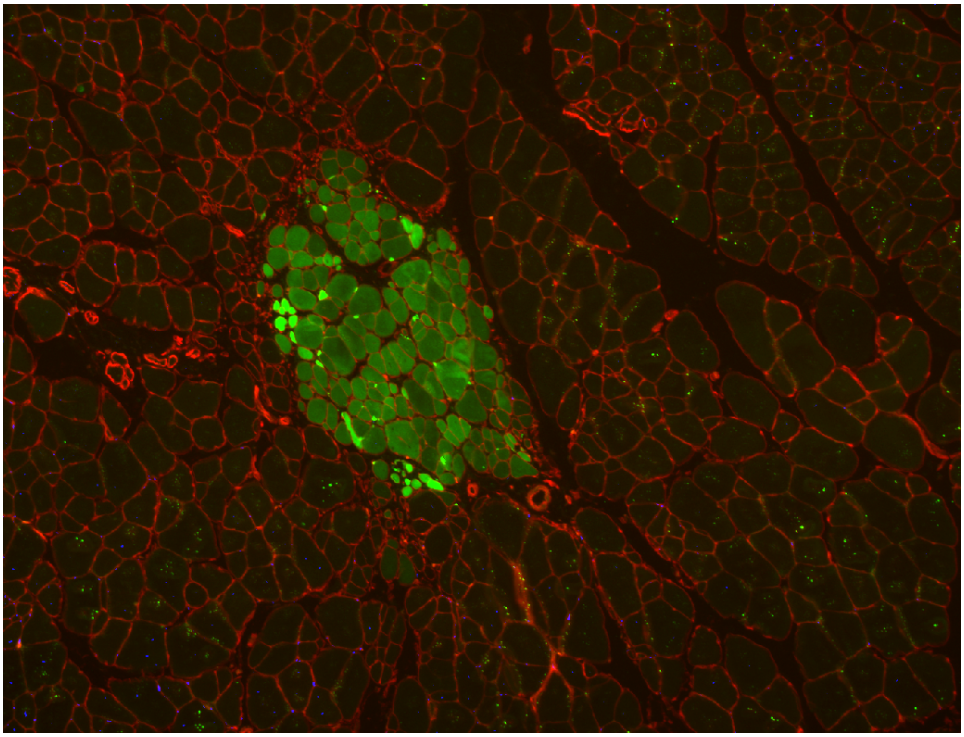


Credit: *Penney Gilbert, Stanford University*

Figure 16: MuSCs in a 600 μm microwell, imaged using 10$\times$ bright field microscopy.

not matter what other channels have been included in the experiment. Figure 17 shows an example of a mouse muscle section stained for laminin, GFP and DAPI, that has been analyzed successfully using the program.

To analyze muscle sections, you first need to load the appropriate settings by clicking **Settings/Load Settings** and loading either *Fibers_low_background.csv* or *Fibers_high_background.csv*. *Fibers_low_background.csv* is for images with low background, where the cracks between the fibers have low intensity in all imaged channels. If it is hard to distinguish cracks from fibers, based only on pixel intensities, *Fibers_high_background.csv* should be used. In this case, you may have to remove cracks between fibers, using the manual correction user interface described below, after the images have been processed. Once you have loaded the settings, you will need to specify the settings "channelNames" and "channelTags", as described in Section 7.6, so



Credit: *Foteini Mourkioti, Stanford University*

Figure 17: A mouse muscle histology section stained for laminin, GFP and DAPI.

that the program knows how to find the channel with the cell membrane staining.

Before you start processing the images, you need to check that the fibers will be segmented correctly, by opening the segmentation GUI described in Section 10.1. If you are not happy with the segmentation results, you can try changing the settings "FibSegBgThreshold" and "FibSegMergeThreshold" described in Table 4. Also make sure that the segmentation setting "SegChannel" is set to the channel with the cell membrane staining.

There is a manual correction user interface designed specifically for correction of muscle fiber segmentations. The user interface is similar to that described in Section 12, and it is opened using the menu option **Manual/ Fiber Correction**. The actual analysis is performed in a user interface which is opened by pressing **Analysis/Fiber Analysis GUI**.

16.3 Myotube fusion anlaysis

In muscle research it is common to perform a fusion assay where myoblasts are grown in differentiation media, to see how well the cells fuse into my-

otubes. Figure 18a shows an image from such an experiment, where red marks myosin heavy chain in the cells and blue marks DAPI in the nuclei. The readout from this assay is usually a parameter called the fusion index. The fusion index is the number of nuclei inside myotubes, divided by the total number of nuclei. A myotube is defined as a red region that contains more than one nucleus.

In order to perform automated analysis of the fusion index, you first need to create segmentation results for the cells in the red channel and the nuclei in the blue channel. Default settings for segmentation of cells and nuclei can be loaded from the files *Fusion_myotubes.csv* and *Fusion_nuclei.csv*, in the settings loading GUI described in Section 7.2. Remember to change the setting "SegChannel" to the desired channel in the segmentation GUI after you have loaded the settings. The segmentation results are created using the tracking GUI found under **Automated/Track**.

Once you have produced segmentations for the cells and the nuclei, and possibly corrected the results in the manual correction GUI described in Section 12, you can open the GUI shown in Figure 19 by pressing **Analysis/Myotube fusion analysis GUI**. In the GUI, you can select which image sequences to process, which tracking (segmentation) versions to use for the cells and the nuclei, and the minimum fraction of a nucleus that has to be covered by a cell region for the nucleus to be considered to be inside the region. When you press **Start**, the program will print out the computed fusion indices for the individual images and the average fusion index across all images. The program will also print the "Pooled fusion index", which is the number of nuclei inside myotubes in all images, divided by the total number of nuclei in all images. If you check the **Plot segmentation results** checkbox before you press **Start**, the program will display images, like the one in Figure 18b, with the outlines of nuclei, myotubes, and unfused myoblasts.

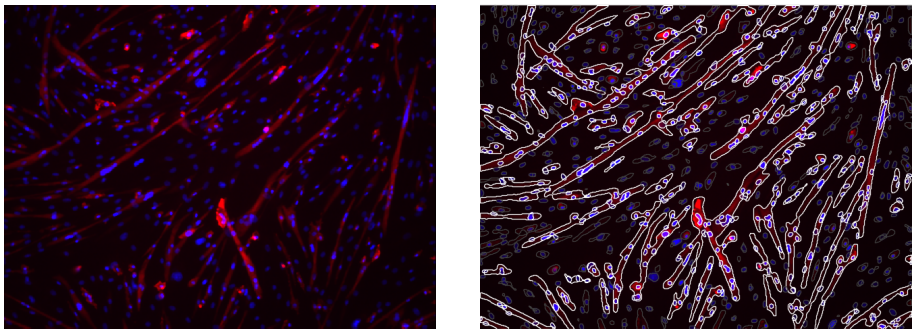
17 Development

If you are a biologist and want to analyze the tracks or outlines of cells in a way currently not possible from within the program, or if you are an algorithm developer and want to modify a processing step in the program, you will need to write some code of your own. This section describes how to write custom analysis scrips and explains the basic structure of the program, to make it easier for researchers to make changes to the program.

17.1 Git

The BA uses Git for version control. The git software can be installed by following the instructions in Section 17.1.1. A Git repository with the version controlled source code is publicly available on `https://github.com/`

Credit: *Antonio Filareto, Stanford University*



(a) (b)

Figure 18: Myoblasts fusing into myotubes (a), where the red channel is myosin heavy chain and the blue channel is DAPI. In the analyzed image (b), myotubes and their nuclei have white outlines and unfused myoblasts and their nuclei have gray outlines. This particular image has a fusion index of 0.54, according to the automated analysis

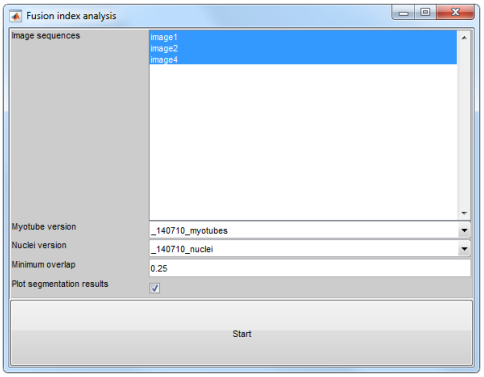


Figure 19: GUI which computes the fusion index of myoblasts that are fusing into myotubes.

`klasma/BaxterAlgorithms` and can be downloaded by following the steps in Section 17.1.2. Once you have downloaded the repository, you can update the source code to the latest version by following the instructions in Section 17.1.3.

17.1.1 Installing Git

Git can be downloaded from `http://git-scm.com` and information on how to use it can be found at `http://gitref.org`. Use the default installation options unless you know what the other options do. On Windows, Git

commands can be executed in a program called Git Bash and there is also a graphical user interface called Git GUI. On Mac, Git commands can be executed in the terminal window and Git GUI can be started by typing **git gui** in the terminal window.

17.1.2 Downloading the BA using Git

Install Git and open Git Bash on Windows or a terminal window on Mac. Go to the directory where you want to put the software folder, using **cd** followed by the directory name. Then type **git clone https://github.com/klasma/BaxterAlgorithms.git** to make a copy of the BA repository on your computer. You can paste text into Git Bash using **Shift+Insert**. The downloaded project is a normal folder containing the program files, but it also contains a hidden folder named *.git*, where all information connected to version handling is stored.

17.1.3 Updating the BA using Git

To update the BA, you do not need to clone the project again. Instead, you can use the Git command **git pull origin master**. This command, and all other Git commands except **git clone**, must be issued from within the project repository. To go there, you use the command **cd** followed by the path of the repository. If you have edited files that would be overwritten by an update, Git will notify you and tell you what your options are.

17.2 Structure of the repository

The Git repository contains files and folders. The files which are not inside a folder can be executed on their own, while the files in the folders are executed from other files. Table 13 describes the files in the root directory of the repository, and Table 14 describes the folders.

17.3 Structure of the data

The outlines and the tracks of the cells are stored in mat-files in subfolders of the experiment's *Analysis* folder, with names that start with *CellData*. The cell data associated with an image sequence can be loaded using the function **LoadCells**, and saved using the function **SaveCells**. The following subsections explain the three classes, **ImageData**, **Cell**, and **Blob**, which are used to store information about the image sequences, the cells, and the outlines of the cells respectively.

17.3.1 ImageData

The ImageData class is located in *ImageData/ImageData.m* and is used to store information about image sequences.

Table 13: Files in the root directory of the repository.

.gitignore	Specifies a set of files which are excluded from version control.
AnalysisExample.m	Simple example of a custom analysis script.
BaxterAlgorithms.m	Creates the main window from which all other GUIs can be opened.
BaxterAlgorithms.prj	Deployment project which can be used to create a Baxter Algorithms program which can be executed without a MATLAB license. The deployment process is described in Section 17.9.
BaxterAlgorithmsTermial.m	Runs the cell tracking without opening a GUI.
CompileMex.m	Compiles C++ code used by the program into mex-files that can be executed by MATLAB.
LICENSE.md	License file which specifies the terms under which the software may be used.
README.md	Short information about how the software can be used.
startup.m	A script which is executed every time MATLAB is started in the repository. The script adds all of the folders in the repository to the MATLAB path, so that all of the files inside them can be found when MATLAB tries to execute them. The folders are also added to the MATLAB path by *BaxterAlgorithms.m*, *BaxterAlgorithmsTerminal.m*, and *CompileMex.m*, so in general, you do not need to execute *startup.m* manually if MATLAB is not started in the repository. That is however necessary before you deploy the software using *BaxterAlgorithms.prj*.

Table 14: The contents of folders in the root directory of the repository.

Analysis	Functions for data analysis and plotting.
Blob	The class Blob, which represents cell outlines, and functions that operate on the class. See Section 17.3.3 for more information.
Cell	The class Cell, which represents cell tracks, and functions that operate on the class. See Section 17.3.2 for more information.
External	External code and the corresponding license files.
Files	All non-executable files which are used by the program, except this user guide.
GUIs	Most of the code for the graphical user interfaces.
ImageData	The class ImageData, which represents image sequences, and functions that operate on the class. See Section 17.3.1 for more information.
PerformanceEvalution	Code and GUIs which are used to compute the segmentation and tracking performance measures SEG and TRA.
Preprocessing	Functions which are used to preprocess the images before the segmentation algorithms are applied.
ReleaseNotes	Release notes for the BA.
Scores	Code to compute tracking scores for different cell counts in segmented regions, and events such as mitosis and cell death.
Segmentation	Code to find the outlines of cells.
Settings	Code to keep track of and manipulate parameters which are used by the different algorithms.
Tracking	Code to link cell outlines into tracks.
UserGuide	This user guide and the files which were used to create it.
Utilities	Low level help functions.

17.3.2 Cell

The cell class is located in *Cell/Cell.m* and is used to store information about the individual cells. When a cell undergoes mitosis, the program ends that Cell object and creates two new Cell objects for the child cells. The Cell objects have the fields "parent" and "children" which store pointers of the Cell objects of the parent and the children of the cell respectively. The x-, y- and z-coordinates of the cell centroids at different time points are stored in the fields "cx", "cy", and "cz" and the outlines at different time points are stored as an array of Blob objects in the field "blob". The field "firstFrame" specifies the first frame that the cell appears in and the field "lifeTime" specifies the number of frames that the cell appears in.

17.3.3 Blob

A Blob object stores information about a cell outline. For 2D data, the field "image" is a binary sub-image of the smallest possible size, where cell pixels are 1's and background pixels are 0's. The field "boundingBox" is a 4 element array specifying where in the original image the sub-image is located. For 3D data, "image" is a binary 3D array and "boundingBox" is a 6 element array which specifies the location of the 3D array in the original z-stack.

17.4 Documentation

To get more information about a function or a class, you can type **help** or **doc** followed by the name of the function or class in MATLAB's command window. The **help** command displays information directly in the command window while the **doc** command displays a formatted version of the information in a separate window. The m-file of the function or class needs to be on the MATLAB path for the **help** and **doc** commands to work.

17.5 Writing custom analysis scripts

The BA has many built in function for data analysis, but if you want a readout which is not available, you can easily write analysis scrips of your own in MATLAB. The following example script plots the number of cells in an image sequence as a function of time.

```
% Path of the image sequence folder.
seqPath = 'C:/ExperimentFolder/ImageSequenceFolder';

% Object with data about the image sequence.
imData = ImageData(seqPath);
% Load outlines and tracks of cells saved
% in the folder CellData_label.
cells = LoadCells(seqPath, '_label');
```

```
% Remove debris.
cells = AreCells(cells);

% Time points in hours.
t = (0:imData.sequenceLength-1) * imData.dT / 3600;
% The number of cells at each time point.
numberOfCells = zeros(imData.sequenceLength,1);
for i = 1:length(cells)
    ff = cells(i).firstFrame;
    lf = cells(i).lastFrame;
    % Increase the cell count.
    numberOfCells(ff:lf) = numberOfCells(ff:lf) + 1;
end

% Plot the cell counts.
figure
plot(t, numberOfCells, 'LineWidth', 2)
xlabel('time (hours)')
ylabel('number of cells')
ylim([0 16])  % Change the limits on the y-axis.
```

Figure 20 shows the plot produced by the above script, for an image sequence with 14 cells at the end.

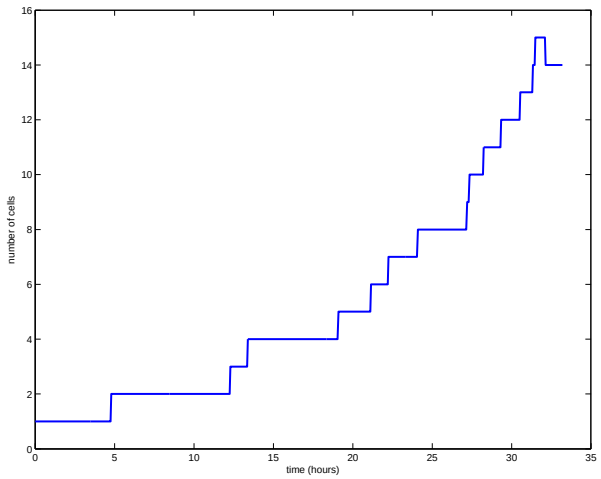


Figure 20: Plot of the number of cells in an image sequence as a function of time.

17.6 Adding a new segmentation algorithm

If you cannot find a good segmentation algorithm in the segmentation GUI described in Section 10.1, you can add a segmentation algorithm of your own to the program. To do this, you need to write a MATLAB function that takes a grayscale image as input and outputs a binary segmentation mask where foreground pixels are 1's. Then you need to add the function to the files *Segmentation/Segment_generic.m* and *Settings/AllSettings.m*.

17.7 External components

The BA uses a number of functions that have been developed by others. The functions and their respective licenses are listed in Table 15. The external code and the licenses can be found in the directory *External* in the root directory of the Git repository. The contents of the license files can also be viewed in Section 22. In cases where modifications have been made, the modifications are released under the MIT license as specified in the license file in the root directory of the Git repository.

17.8 Compiled C++ code

The program contains some compiled C++ functions. The functions have been compiled for 64-bit Windows and 64-bit Mac, but if you are using a different operating system or if you make changes to the C++ code, you will need to compile the functions into mex-files. This is done using the function **CompileMex**.

In order to compile mex files with **CompileMex**, you need to have a C++ compiler installed and tell MATLAB to use the compiler by running the command **mex -setup**. On Mac you can get a C++ compiler by installing Xcode from `https://developer.apple.com/xcode/`. In order to use the compiler, you may also need to install command line tools for Xcode, which can be done from within Xcode. On Windows, you can get a C++ compiler by installing Visual Studio Express.

17.9 Deploying the BA

Users who do not have a MATLAB license can use a deployed version of the BA. Deployed versions can be downloaded and installed for 64-bit Windows and Mac, by following the instructions in Section 4.1. For other operating systems, there are no deployed versions available. To create a deployed version, you need to have a MATLAB Compiler license and have MATLAB installed, together with the toolboxes needed by the BA, on the type of operating system that you want to deploy for. Before you deploy the BA, you need to make sure that all program folders are on the MATLAB path by opening the root directory of the program and running *setup.m*. To deploy

Table 15: External code components written by others.

Name	License	Description
bwdistsc	BSD (2 clause)	Computes Euclidean distance transforms of binary 3D images with non-isotropic voxel aspect ratios. Copyright 2007 Yuriy Mishchenko.
getGitInfo	BSD (2 clause)	Returns information about the current branch and commit in a Git repository. The inputs and outputs have been modified. Copyright 2011 Andrew Leifer.
medfilt3	BSD (2 clause)	Performs median filtering of 3D arrays. Copyright 2011 Simon Robinson and 2014 Jimmy Shen.
ParforProgMon	BSD (3 clause)	A progress bar which can be used in a parfor loop. Copyright 2009 The MathWorks, Inc. and 2011 Willem-Jan de Goeij.
plotboxpos	MIT	Returns the position of an axes object inside a figure. Copyright 2015 Kelly Kearney.
uigetfile_n_dir	BSD (2 clause)	The original function is used to select multiple files and directories. A modified version which only selects multiple directories is used in the BA. Copyright 2011 Peugas.

the program, you open the file *BaxterAlgorithms.prj* by double clicking it in MATLAB. Then you press the **Package** button. This creates an installer file in a new folder named *BaxterAlgorithms/for_redistribution*. On 64-bit Windows, the file is called *BaxterAlgorithmsInstaller_win64.exe*, and on 64-bit Mac it is called *BaxterAlgorithmsInstaller_maci64.app*. When the installer is executed, it installs the BA and also downloads and installs the MATLAB Compiler Runtime if it is not already installed. The MATLAB Compiler Runtime is a free software which can run deployed MATLAB programs.

18 How to cite

If you use the software for a biological study, please site [1], which describes the software in detail. Use [4] to cite the track linking algorithm used by the BA, and use [7] if you want a citation describing the performance of the BA in the Cell Tracking Challenges.

19 Solving problems

Bugs related to unexpected inputs or user operations can sometimes cause problems in the program and sometimes the program will not work properly afterwards. To recover from such problems, first try restarting the program and then try restarting MATLAB.

20 Support

Please post bug reports, feature requests, and enhancement requests on `https://github.com/klasma/BaxterAlgorithms/issues`. When you report a bug, please make sure that you include the MATLAB version, the operating system, the MATLAB error message, and step by step instructions on how to reproduce the bug. Also say whether or not you are running a deployed version of the software.

If you have a question that you cannot find the answer to in this user guide, you can email it to klasmagnus@gmail.com.

21 Acknowledgements

The software was developed by Klas Magnusson in a collaboration between the Department of Signal Processing at KTH in Sweden and the Blau Lab at Stanford University in California, USA. Klas was supervised by Joakim Jaldén at KTH and Helen Blau at Stanford University. In the beginning of the project, Klas was advised by Sebastian Thrun at Stanford University.

The idea to automate cell tracking in the Blau Lab came from Karen Havenstrite, and together with Penney Gilbert, she was the first to use an early version of the software in a biological study. The idea to automate the analysis of fluorescently labeled histological sections of muscle came from Andrew Ho in the Blau Lab. The software is named after the Baxter International Foundation, which provided initial funding to start the project. Code contributions have been made by Andrew Chan, who improved graphical user interfaces, and by Colin Holbrook, who made enhancements based on his own user experience. Many others have made important contributions by testing the software and applying it to new types of data.

22 Licenses

This Section contains the contents of all license files, so that they are not separated from the software when it is released in binary form. The BA is released under the MIT license. The license file is located in the root directory of the Git repository. Section 22.1 shows the contents of the license file. The contents of the license files of the external components described in Section 17.7 are shown in Sections 22.2 - 22.7. The actual license files can be found in the respective subfolders in the *External* directory in the root directory of the Git repository. Note that all modifications to the external components are released under the MIT license.

22.1 License for BA

```
MIT License

Copyright (c) 2018 Klas Magnusson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

22.2 License for bwdistsc

```
Copyright (c) 2007, Yuriy Mishchenko
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

22.3 License for getGitInfo

```
Copyright (c) 2011, Andrew
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

22.4 License for medfilt3

```
Copyright (c) 2011, Simon Robinson
Copyright (c) 2014, Jimmy Shen
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

22.5 License for ParforProgMon

```
Copyright (c) 2011, Willem-Jan de Goeij
Copyright (c) 2009, The MathWorks, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the distribution
    * Neither the name of the The MathWorks, Inc. nor the names
      of its contributors may be used to endorse or promote products derived
      from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
```

INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

22.6 License for plotboxpos

The MIT License (MIT)

Copyright (c) 2015 Kelly Kearney

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

22.7 License for uigetfile_n_dir

Copyright (c) 2011, Peugas
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE

POSSIBILITY OF SUCH DAMAGE.

References

[1] K. E. G. Magnusson, *Segmentation and tracking of cells and particles in time-lapse microscopy*, Ph.D. thesis, KTH Royal Institute of Technology, 2016.

[2] P. M. Gilbert, K. L. Havenstrite, K. E. G. Magnusson, A. Sacco, N. A. Leonardi, P. Kraft, N. K. Nguyen, S. Thrun, M. P. Lutolf, and H. M. Blau, "Substrate elasticity regulates skeletal muscle stem cell self-renewal in culture," *Science*, vol. 329, no. 5995, pp. 1078–1081, 2010.

[3] K. E. G. Magnusson and J. Jaldén, "A batch algorithm using iterative application of the Viterbi algorithm to track cells and construct cell lineages," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2012, pp. 382–385.

[4] K. E. G. Magnusson, J. Jaldén, P. M. Gilbert, and H. M. Blau, "Global linking of cell tracks using the Viterbi algorithm," *IEEE Trans. Med. Imag.*, vol. 34, no. 4, pp. 1–19, 2015.

[5] M. Maška, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. España, S. Venkatesan, D. M. W. Balak, P. Karas, T. Bolcková, M. Štreitová, C. Carthel, S. Coraluppi, N. Harder, K. Rohr, K. E. G. Magnusson, J. Jaldén, H. M. Blau, O. Dzyubachyk, P. Křížek, G. M. Hagen, D. Pastor-Escuredo, D. Jimenez-Carretero, M. J. Ledesma-Carbayo, A. Muñoz-Barrutia, E. Meijering, M. Kozubek, and C. Ortiz-de-Solorzano, "A benchmark for comparison of cell tracking algorithms," *Bioinformatics*, vol. 30, no. 11, pp. 1609–1617, 2014.

[6] "Cell Tracking Challenge," `http://celltrackingchallenge.net`, Accessed: 2018-12-10.

[7] V. Ulman, M. Maška, K. E. G. Magnusson, O. Ronneberger, C. Haubold, N. Harder, P. Matula, P. Matula, D. Svoboda, M. Radojevic, I. Smal, K. Rohr, J. Jaldén, H. M. Blau, O. Dzyubachyk, B. Lelieveldt, P. Xiao, Y. Li, S.-Y. Cho, A. C. Dufour, J.-C. Olivo-Marin, C. C. Reyes-Aldasoro, J. A. Solis-Lemus, R. Bensch, T. Brox, J. Stegmaier, R. Mikut, S. Wolf, F. A. Hamprecht, T. Esteves, P. Quelhas, Ö. Demirel, L. Malmström, F. Jug, P. Tomancak, E. Meijering, A. Muñoz-Barrutia, M. Kozubek, and C. Ortiz-de-Solorzano, "An objective comparison of cell-tracking algorithms," *Nature methods*, vol. 14, no. 12, pp. 1141–1152, 2017.

[8] K. Li, "The image stabilizer plugin for ImageJ," `http://www.cs.cmu.edu/~kangli/code/Image_Stabilizer.html`, Feb. 2008.

[9] P. Matula, M. Maška, D. V. Sorokin, P. Matula, C. Ortiz-de-Solórzano, and M. Kozubek, "Cell tracking accuracy measurement based on comparison of acyclic oriented graphs," *PLOS ONE*, vol. 10, no. 12, pp. 1–19, 2015.