Chapter 19

A Cell Segmentation/Tracking Tool Based on Machine Learning

Heather S. Deter, Marta Dies, Courtney C. Cameron, Nicholas C. Butzin, and Javier Buceta

Abstract

The ability to gain quantifiable, single-cell data from time-lapse microscopy images is dependent upon cell segmentation and tracking. Here, we present a detailed protocol for obtaining quality time-lapse movies and introduce a method to identify (segment) and track cells based on machine learning techniques (*Fiji's Trainable Weka Segmentation*) and custom, open-source *Python* scripts. To provide a hands-on experience, we provide datasets obtained using the aforementioned protocol.

Key words Computational image analysis, Single-cell quantification, Cell lineage analysis, Cell segmentation, Cell tracking, Machine learning, Fluorescence microscopy, Bacterial growth

1 Introduction

During the last decade, there has been a transition in the analysis of cellular physiology from the batch level (population-scale) to the single-cell level. This transition has been stimulated by the development of quantitative and high-throughput techniques that require computer-aided methods to extract information at the single-cell level. Using these approaches researchers have been able to show, for example, the relevance of stochastic sources in gene expression [1] or the logic underlying cell size homeostasis [2]. Thus, single-cell analysis is less subject to averaging effects (Fig. 1) and offers a level of discrete detection that is unobtainable with traditional techniques [3–6]. In this context, the adoption of single-cell microscopy techniques has been limited because identifying, tracking, and quantifying single cells within a population of cells are usually difficult, time-consuming, and prone to errors that

Electronic supplementary material: The online version of this chapter (https://doi.org/10.1007/978-1-4939-9686-5_19) contains supplementary material, which is available to authorized users.

Elena Rebollo and Manel Bosch (eds.), *Computer Optimized Microscopy: Methods and Protocols*, Methods in Molecular Biology, vol. 2040, https://doi.org/10.1007/978-1-4939-9686-5_19,

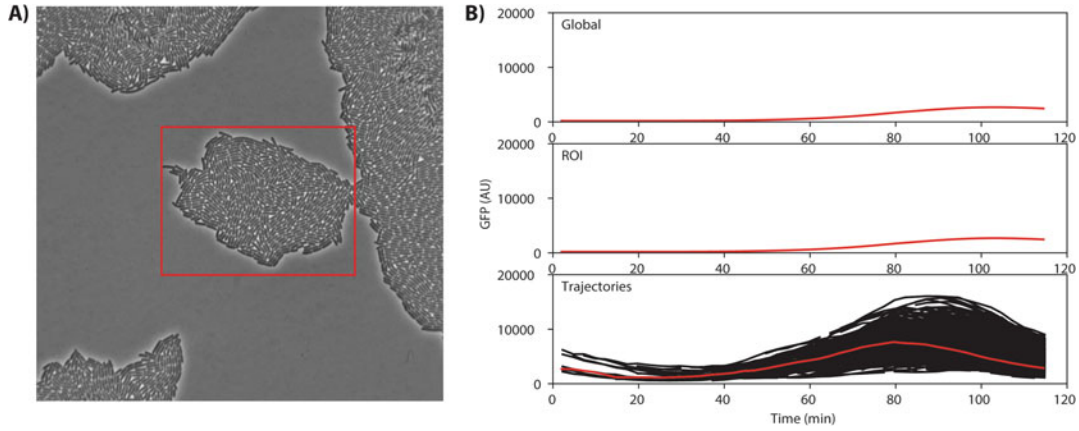

Fig. 1 (**a**) The final frame of the image dataset. The region of interest (ROI) is outlined in red. (**b**) Median fluorescence for given selections over time (below). Global: the median fluorescence over time for the whole image. ROI: the median fluorescence over time for the ROI is outlined in red in (**a**). Trajectories: the fluorescence over time for each trajectory (black) and the median fluorescence for all cells (red)

require manual corrections. Indeed, the identification stage implies a methodology able to recognize and outline the domain of individual entities (segmentation) and is particularly critical since tracking and quantification depend on it.

Traditional cell segmentation algorithms are based on image-processing techniques that ultimately compute gradients and use thresholding to measure the intensity and spatial relationships of pixels in order to detect cell boundaries. The latter is especially challenging in dense cell populations, e.g., bacterial colonies, and, while some edge detectors have been proven to be more effective than others, e.g., *Marr-Hildreth* vs. *Canny* [7], small changes in the microscopy illumination conditions require, more often than not, nontrivial adjustments of the segmentation parameters. In that regard, during the last years, a number of segmentation/tracking software suites have been publicly released [8–10]. Here we highlight three examples that, while essentially based on, and consequently constrained by, the aforementioned methodology, stand out because of their additional features and reliability. *MicrobeJ* [11, 12] is a plugin available through *Fiji/ImageJ* [13] that has a wide variety of tools available to analyze cell morphology and track cells in their user-friendly interface. *Oufti* [14] offers a user-friendly interface and a number of functionalities for quantitative analysis that include subpixel resolution for "reading" fluorescent signals within single cells. On the other hand, *CellX* [15] uses a novel approach for cell segmentation based on membrane patterning that is versatile in terms of cell shapes and robust to image noise.

More recently, the advent of artificial intelligence and machine learning techniques into the field has made possible the development of segmentation/tracking tools able to learn from training

datasets and improve from experience without the need of explicit programming or parameter tweaking, e.g., *CellProfiler* [16] or more recently *SuperSegger* [17]. Here, following these ideas, we present a detailed protocol that utilizes an open-source *Fiji/ImageJ* plugin [13], the *Trainable Weka Segmentation tool* [18], complemented by custom-made open-source *Python* scripts. The computational methods herein can be used to count and track objects in any series of 16-bit tiff images. We have used these methods to count colonies on agar plates and track cells in microscope images. Here we detail one method of obtaining microscope images, which aims to reduce the training queue and improve the segmentation/tracking process. To give a hands-on experience, we provide a dataset in the context of bacterial growth that was obtained using this method [19]. We have also recorded a video tutorial (Video 1) available on YouTube [20] and in the data repository to assist the users.

2 Materials

2.1 Reagents

1. 5× A Salts (composition per 100 mL): 0.046 g $(NH_4)_2SO_4$, 2.25 g KH_2PO_4, 5.25 g K_2HPO_4, 0.25 g sodium citrate tribasic·$2H_2O$, and 100 mL sterile deionized water [21]. Proceed to filter sterilize.
2. A minimal medium (composition per 100 mL): 20 mL (5×) A Salts, 80 mL sterile deionized water, 100 μL (1 M) $MgSO_4 \cdot 7H_2O$, 250 μL (80%) glycerol, 0.4% (w/v) glucose, and 1 mL (10%) casamino acids [21].
3. Lysogeny Broth Miller (LB; *see* list of abbreviations in Supplementary Table S1).
4. 100 mM Isopropyl β-D-1-thiogalactopyranoside (IPTG).
5. 50 mg/mL Kanamycin.
6. Silica gel.
7. Low melting agarose (2-hydroxyethyl agarose, Sigma).

2.2 Equipment

1. Two coverslips 24 × 60 mm, thickness No. 1.
2. One biopsy punch 6 mm diameter.
3. Desiccator (or an airtight container that can be used as a desiccator).
4. Microwave.
5. Parafilm.
6. A 96 well plate lid or an equivalent glass surface.
7. A 0.2 μm pore-size filter.

8. A 10 mL syringe (in which the 0.2 μm pore-size filter can be coupled).
9. Scalpel.
10. IBIDI μ-dish 35 mm low (bottom: polymer coverslip No. 1.5, thickness: 180 μm).
11. Microscope/imaging equipment: we used an inverted microscope (Leica DMi8) equipped with a 100×/1.40 oil objective (HC PL APO, Leica), Köhler illumination, a CMOS camera (Hamamatsu ORCA-Flash4.0 camera, V2), and green fluorescent protein (GFP; Ex: 470/40 nm, Em: 525/50 nm) filter. Excitation was performed using a led lamp (Lumencor light engine SOLA SE 5-LCR-VA). Leica Application Suite X (LAS X) software was used to control microscope and acquire images using Live Data Mode.
12. Temperature control system: IBIDI heating system. All experiments are performed at 37 °C.

2.3 Operating System, Software, and Data Repository

All of the following are open-source, and downloads are available online (*see* Tables 1, 2, and Supplementary Table S2).

1. *Ubuntu 16.04 LTS*, a Linux operating system. Alternatively, Mac OS X can also be used to run the pipeline. To run the pipeline on other operating systems, a virtual machine [22] can be installed to use *Ubuntu* (*see* **Note 1**).
2. *Fiji/ImageJ*, an open-source *Java* image-processing program [13].
3. *Anaconda 2.7* is an open-source distribution of *Python*, a programming language that has a wide range of tools and libraries for image analysis, including *SciPy* and *NumPy* (*see* **Note 2**). Our scripts have exclusively been tested with *Python* 2.7.
4. *OpenCV* is a *Python* package required for the pipeline that is not included in the initial *Anaconda* download (Table 2). Only *OpenCV* downloaded through *Anaconda*, using the command *conda install opencv* in the terminal, has been tested to work with our scripts (*see* **Note 3**).
5. *Avconv*, a library for video and audio conversion. This library can be installed in *Ubuntu* using the following command in the terminal: *sudo apt install libav-tools*. To install *Avconv* for Mac OS X, the command is *brew install libav* (*Homebrew* must be installed prior to installing *Avconv*).
6. A list of scripts is available in Table 2. Scripts must be run in an *Anaconda Python* 2.7 environment with *OpenCV* installed to function as designed. The Master Script, *SegmentandTrack.py*, calls the remaining scripts to run the entire analysis pipeline based on user input. All scripts and imaging datasets are available at the OSF public repository [19]. These images can be

Table 1
List of all software and operating systems

Software	Function	Website
Ubuntu 16.04 LTS	Linux operating system	https://www.ubuntu.com/download [28]
VirtualBox (optional)	Virtual machine to run Linux environment	https://www.virtualbox.org/wiki/Downloads [22]
Fiji ImageJ (includes Weka segmentation tool)	Classification of images using machine learning	https://imagej.net/Fiji/Downloads [13, 23]
Anaconda (Python 2.7)	Open-source distribution of Python and related packages (including NumPy and SciPy)	https://www.anaconda.com/download/ [29]
OpenCV	A Python package with tools for image analysis (not included in Anaconda). On a Linux machine, install through Anaconda using the command "conda install opencv" in the terminal	https://pypi.python.org/pypi/opencv-python [30]
Avconv	Software package for handling videos	https://libav.org/avconv.html#Description [31]

used to recreate Videos 2–4. Figure 2 demonstrates how an image was segmented prior to cell tracking.

7. 16-bit tiff images. Sample image datasets are available in the data repository [19]. These datasets contain 467 frames and 20 frames.
8. The recommended hardware to process the complete imaging dataset (467 frames) is 8 GB or more of RAM and a modern processor. An image subset (20 frames) requiring a less demanding hardware configuration is also available in the data repository. Adjustments in *Fiji* memory settings may be required for processing the dataset (the *Fiji* software requires at least 1 GB RAM for its processes [23]).

3 Methods

Here we present one method of obtaining images, followed by an image-processing pipeline that can be applied to a variety of image datasets. The processing pipeline consists of a series of steps that were designed to analyze a swath of single-cell datasets (Table 2). These scripts are designed to work on 16-bit tiff images that use a file naming system containing the letters "p" or "g" to indicate phase or fluorescence, respectively, followed by three numbers to

Table 2
List of custom scripts

Script	Subheadings	Language	Brief description	CSV data
SegmentandTrack.py	3	Python	The master script to run the pipeline based on user input	N/A[a]*
Image_alignment.py	3.4	Python	Aligns images based on differences calculated through FFT	N/A
Segmentation.ijm	3.5	Fiji macro (ijm)	Calls Trainable Weka Segmentation tool and can be used to train or apply classifiers	N/A
Batch_segment.bsh	3.5	BeanShell	Called by RunWeka.py to segment a batch of images	N/A
RunWeka.py	3.5	Python	Calls Segmentation.ijm and Batch_segment.bsh	N/A
TrackCellLineages.pyCellLineages	3.6	Python	Labels a binary mask and calculates the differences between a given cell and cells within a given area in the previous image. Automatically saves single-cell data. Uses the calculated differences to find trajectories and identify cell lineages. Labels lineages from the first frame and outputs lineage data.	Single-cell data and lineage data summary
Lineage_analysis.py	3.7	Python	Outputs csv files with frame-by-frame data for lineages tracked in TrackCellLineages.py	Lineage data for individual lineages
Image_analysis.py	3.7 and 3.8	Python	Analyzes global or ROI data	Global and ROI data

All scripts can be modified and include comments to facilitate modification. Scripts can be downloaded at GitHub https://github.com/hdeter/CellTracking [32] or at the public repository http://osf.io/gdxen/ [19]
[a]CSV files output by other scripts will also be output when running SegmentandTrack.py

indicate frame and then the extension ".tif" (e.g., 20171212_book-**p-001.tif**, indicating **p**hase contrast channel and frame number **001**). Phase and fluorescence images must be separate (not stacked). Modification of the scripts is required to use alternative naming systems, and some modification is required for datasets containing two or more fluorescence channels (*see* **Note 4**).

3.1 Sample Preparation

The *E. coli* strain used for imaging was MG1655 transformed with a ColE1 plasmid containing resistance to Kanamycin and the inducible combinatorial promoter $P_{lac/ara-1}$ [24] controlling the expression of GFP marked with an LAA degradation tag. We broke the sample preparation down to three major steps:

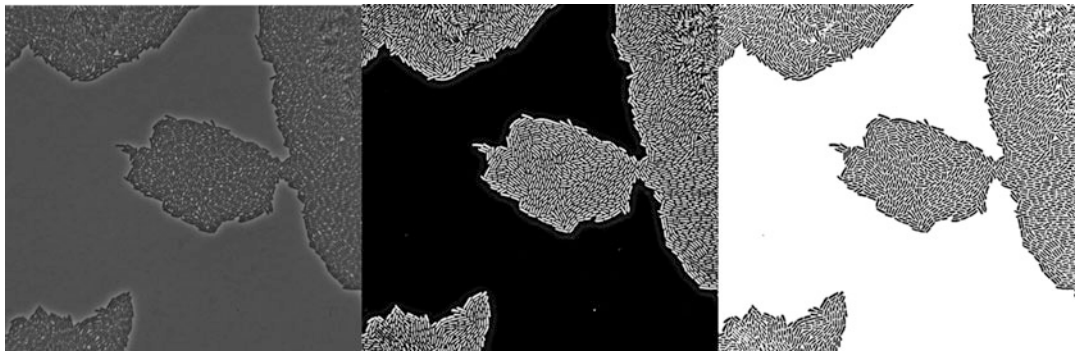

Fig. 2 An example of cell segmentation using our method. Left: phase image. Center: Mask 1, a probability mask based on the classification of the phase image. Right: Mask 2, a binary mask based on the classification of Mask 1

1. Grow cells overnight from a glycerol stock in 10 mL LB with 10 μL (50 mg/μL) Kanamycin at 37 °C, shaking at 200 rpm.
2. Dilute cells to a final OD_{600} ~0.01 in 5 mL A minimal media (*see* **Note 5**) with 1.5 μL (100 mM) IPTG, to achieve a final concentration of inductors of 0.03 mM IPTG.
3. Let this culture grow at 37 °C and 200 rpm until OD_{600} is between 0.2 and 0.3 (~3 h). While cells are growing, prepare agarose pads (**steps 4–8**).
4. Cover a 96 well plate lid with parafilm, extending the parafilm over its surface. Place one coverslip in the middle of the parafilmed area, and gently press the cover glass' four corners against the parafilm to "fix" it.
5. Mix 0.2 g low melting agarose (*see* **Note 6**) in 10 mL A minimal medium, and go through three to four cycles of heating the agarose solution in a microwave until it starts to boil and vortexing the mixture.
6. Once a homogeneous agarose solution is ready and it has cooled down to around 50 °C, add the necessary inducers/antibiotics. In our case, we added 3 μL (100 mM) IPTG. Vortex the mixture gently, and filter it immediately with a 0.2 μm pore-size filter using the 10 mL syringe (*see* **Note 7**).
7. Allow the mixture to sit (for a few minutes) so that bubbles accumulate by the surface (*see* **Note 8**). Pipette around 3 mL of the agarose mixture from the bottom part of the tube (bubble-free) onto the cover glass slide. Immediately place another coverslip on top of the poured agarose to create an agarose *sandwich*. Take special care to minimize introduction bubbles while preparing the sandwich (first make a gentle contact between the upper cover glass and the agarose in one of the extremes, and then proceed to lower down the rest of the cover glass).

8. Let the pad dry at room temperature for at least 1 h before spotting cells onto it (*see* Subheading 3.1, **step 13**). Note that if the pad is left to dry "too much," it will shrink and will not be useful/reliable. To prevent excessive evaporation from edges, cover the agarose sandwich with a lid. Additional details can be found here [25].
9. Once cells have grown between OD_{600} 0.2 and 0.3, dilute them to a final concentration of ~0.01 OD_{600} using A minimal media with 0.03 mM IPTG to a final volume of 1 mL.
10. Carefully slide/detach the upper cover glass from the agarose sandwich, and cut a smaller pad using a 6 mm diameter biopsy punch (*see* **Note 9**).
11. Spot 2 μL of the diluted culture ("seed the pad") from **step 2** onto this smaller pad.
12. Incubate the pad at 37 °C for 15 min, and then place it in a desiccator (containing silica gel) at room temperature for another 15 min. This drying process of the seeded pad should avoid any drift in the imaging (*see* **Note 10**).
13. After drying, place the prepared pad in a cover glass bottom dish so that the seeded surface is in direct contact with the bottom of the dish (use a scalpel to do this, and avoid touching the seeded surface of the pad).
14. Seal the dish with parafilm to prevent pad shrinkage due to evaporation. The sample is now ready for imaging.

3.2 Imaging

1. Set up the microscope for Köhler illumination. A precise alignment of the optical components in the optical path (including the phase contrast ring) is a critical step for getting quality images and, as a consequence, a reliable segmentation. The procedure to do this is outside the scope of this manuscript, and there are excellent interactive tutorials on the web showing how to achieve alignment, including one available from Nikon [26].
2. To generate our image dataset, we used the following time-lapse sampling times: phase images were taken every 30 s and fluorescence images of GFP were taken every 5 min (*see* **Note 11**). We set 15 and 30 ms exposure time (using the maximum intensity of the lamp) for phase contrast and GFP channels, respectively. These conditions are experiment and equipment dependent.

3.3 Running the Microscope Image Analysis Pipeline

We have developed specific custom scripts that utilize open-source software (*see* Subheading 2.3) for cell segmentation and lineage tracking. To facilitate use, we provide *SegmentandTrack.py*, a Master Script, to run the entire pipeline based on user input (Tables 2 and 3). The pipeline (*see* Table 2) has been tested using an *Ubuntu*

Table 3
List of prompts for user input by SegmentandTrack.py

Prompt	Description	Subheadings
Do you wish to align images? (Y/N)	Answer "y" to align images	3.5
Do you wish to train and/or apply a classifier? (Y/N):	Answer "y" to segment images	3.6
Do you wish to track cells? (Y/N)	Answer "y" to track cells	3.7
Do you wish to output csv files detailing data for individual lineages? (Y/N)	Answer "y" to output frame-by-frame data for individual lineages as a csv file	3.7
Do you wish to analyze images (needed to get whole image fluorescence or render videos)? (Y/N)	Answer "y" to analyze the entire image and render videos	3.8
Do you wish to render videos? (Y/N)	Answer "y" to output videos	3.9
Enter the name of the image directory, relative to the working directory (e.g., Practice)	Enter the name of the directory containing your images (must be in your working directory)	3.3
Enter the number of the first frame in the dataset (e.g., 448)	Enter a whole digit integer corresponding to the first image in the dataset	3
Enter the number of the last frame in the dataset (e.g., 467)	Enter a whole digit integer corresponding to the last image in the dataset	3
Enter the time per frame in minutes (e.g., 0.5)	Enter the time in minutes (e.g., 30 s is 0.5 min)	3.2
Enter the number of the first frame with a fluorescence image (e.g., 449; for no fluorescence enter 0)	Enter a whole digit integer corresponding to *the first fluorescence image* in the dataset	3
Enter the number of frames between fluorescence images (i.e., every nth image; for no fluorescence enter 0)	Enter how often fluorescence images occur (e.g., every nth frame)	3.2
Enter the images' file name preceding the channel and file number (e.g., 20171212_book)	Enter the portion of the file name that does not change (i.e., precedes channel and file number)	3
Enter the name of fluorescence channel 1 (e.g., GFP)	Enter the name of the fluorescence channel	3.2
Do you have an ROI file for a stationary area? (Y/N)	Answer "y" if using an ROI for image alignment	3.4 and 3.5
Enter the path to the csv file, relative to the working directory (e.g., Align_roi.csv)	Enter the path to the csv file containing the ROI to use for image alignment	3.4 and 3.5
Enter the name of the directory to output images into (e.g., Aligned)	Enter a name to call the directory to which images will be output	3.5
Do you have masks for the images? (Y/N)	If not segmenting images, answer whether or not you have binary masks	3.6

(continued)

Table 3
(continued)

Prompt	Description	Subheadings
Enter the name of the directory containing masks, relative to [image directory]	If you have binary masks, enter the directory containing them, which must be within your image directory	3.6
Enter the absolute path to the Fiji executable file (e.g., /home/user/Downloads/Fiji.app/ImageJ-linux64)	Enter the full path to the executable file, located in the Fiji.app folder on your computer	3.6
Is there an open instance of Fiji? (Y/N)	Answer "y" once Fiji is open	3.6
How many rounds of classification are you running? (1 or 2)	Determines how many rounds of segmentation to run; we recommend two rounds	3.6
Would you like to batch classify the images in the background? (Y/N)	Answer "y" for background classification (only available on Linux), which allows multiple images to be segmented simultaneously	3.6
Enter how many processes are available to use for multiprocessing; set to 1 for no multiprocessing:	If classifying images in the background, enter the number of processes (image classifications) that can be run simultaneously. Check the number of threads available on your specific computer	3.6
Do you have a trained classifier? (Y/N)	Answer "n" if you need to train a classifier in Fiji. If you already have a model file (classifier) saved, answer no	3.6
Enter the path to the classifier, relative to the working directory (e.g., Aligned/classifier.model)	Enter the path to the classifier relative to your working directory (it will likely be saved within your image directory)	3.6
Enter the name of the directory within [image directory] to output masks into (e.g., Mask 1)	Enter a name to call the directory to which images will be output	3.6
Enter the name of the directory containing images to classify, relative to the working directory (e.g., Aligned/Mask 1)	Enter the name of the image directory with the images you wish to segment	3.6
Enter the minimum cell area for tracking (e.g., 100)	Enter a minimum size for cell tracking (area in pixels)	3.7
Enter the maximum cell area for tracking (e.g., 2500)	Enter a maximum size for cell tracking (area in pixels)	3.7
Enter the minimum number of frames to track cells through (e.g., 15)	Enter the minimum number of frames to track cells. Trajectories with a shorter length will be excluded	3.7
Enter the name of the directory to output csv files into, relative to the working directory	Enter a name to call the directory to which the files will be output	3.7

(continued)

Table 3
(continued)

Prompt	Description	Subheadings
Do you wish to get data for all of the lineages? (To only analyze select lineages based on lineage name answer no; Y/N)	Answer "y" to output a csv file for each lineage. Answer "n" to input the names of specific lineages for which you wish to output a csv file	3.7
Enter the number of lineages you wish to analyze	Enter how many lineages you wish to analyze as a whole digit integer (*see* **Note 28**)	3.7
Input the full name for each lineage (e.g., 0001, 0001–2, 10001) Lineage #	For each lineage you wish to analyze, input the name of the lineage and hit enter (*see* **Note 28**)	3.7
Do you wish to analyze a region of interest? (Y/N)	Answer "y" to get fluorescence for an entire region of interest (must have ROI file)	3.8
Do you wish to crop the images based on an ROI? (Y/N)	Answer "y" to crop the images (and final video) to a region of interest (must have ROI file)	3.9
Enter the path to the csv file for the ROI to analyze, relative to the working directory (e.g., ROI.csv)	If analyzing or cropping to an ROI, input the path (relative to the working directory) to the csv file	3.8 and 3.9
Do you want to number the cells in the images based on lineage tracking? (Y/N)	Answer "y" to write cell labels into the final video	3.9
Do you want to contour cells based on masks? (Y/N)	Answer "y" to contour (outline) cells in the final video	3.9

16.04 LTS operating system (recommended) and Mac OS X (*see* **Note 1**). For convenience, place the scripts in a folder that also contains the images in a subfolder. It is essential to read the entire protocol before running the analysis pipeline for optimal operation.

To run *SegmentandTrack.py* (*see* Video 1 for a video tutorial):

1. Open a terminal. Keep in mind that the terminal is case-sensitive.
2. Change directories to the folder that contains the scripts and the images that are to be analyzed (Fig. 3a). Note that directories will differ based on individual machine and file locations (*see* **Note 12**).
3. Type "Python SegmentandTrack.py" (Fig. 3a).
4. Answer the prompts (Table 3; please *see* Subheadings 3.4–3.9 for further explanation). For yes or no (Y/N) questions, answer "y" for yes or "n" for no.

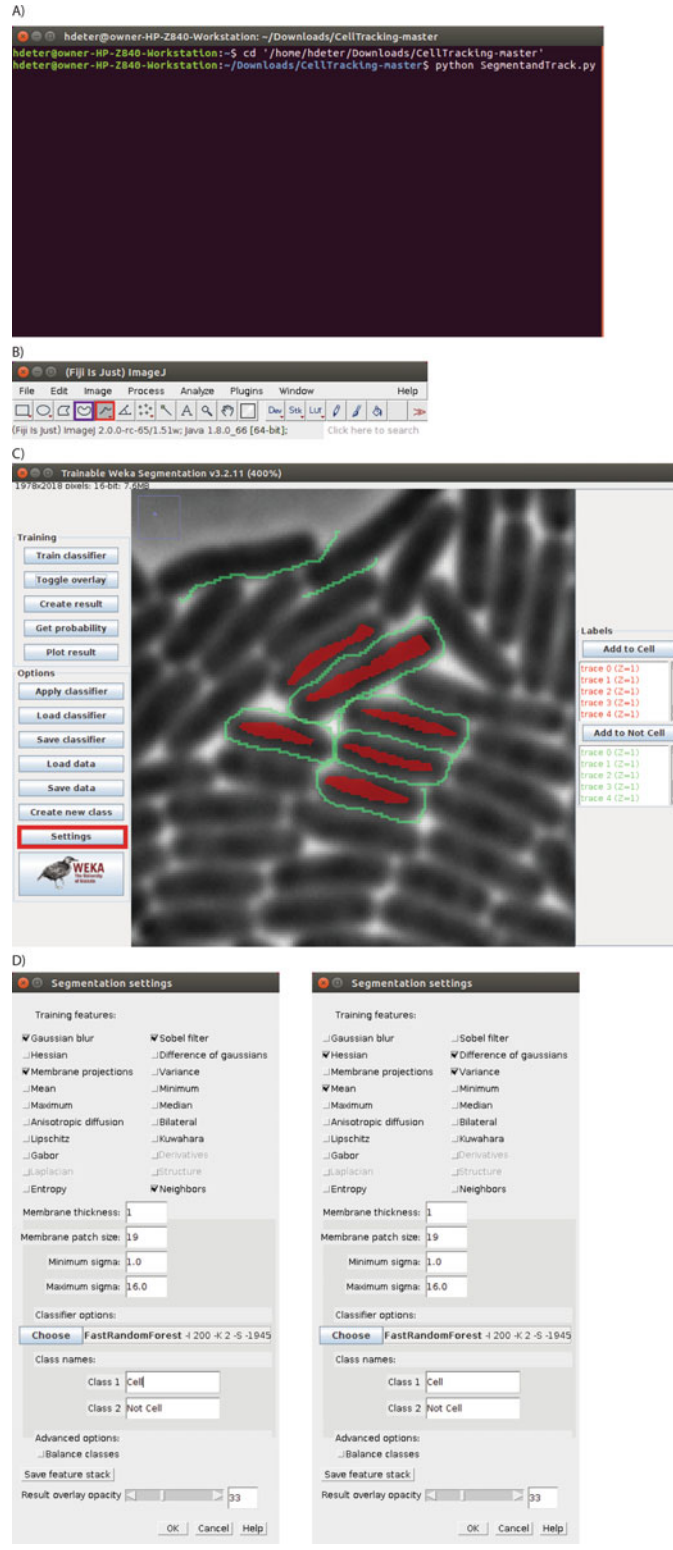

Fig. 3 The *Weka* settings for our classifiers. Here we highlight default settings that have worked well in the past, but these parameters can be altered depending on the user requirements. (**a**) An example of commands to run a

3.4 Selecting a Region of Interest (ROI) (Optional)

Regions of interest (ROIs) can be used to crop images or select a specific region for a particular type of analysis. SegmentandTrack.py offers two options that use an ROI (*see* Subheadings 3.5 and 3.9), which relies on a csv (comma-separated values) file created using the *Fiji* measure tool as follows (*see* Video 1 for a video tutorial):

1. Open the desired image in *Fiji*. To observe the same region throughout multiple images, import an image sequence using [File > Import > Image Sequence...].
2. Remove any scale associated with the images using [Analyze > Set Scale...] (*see* **Note 13**).
3. Go to [Analyze > Set measurements...], and choose *Bounding rectangle* as the only checked box; use 0 decimal places (the results need to be whole, even numbers; *see* **Note 14**).
4. Select the desired region with the rectangular selection tool and use the *measure* tool, located at [Analyze > Measure] (Fig. 3b). Save the results as a *csv* file (e.g., filename.csv) in the same directory as the scripts.

3.5 Image Alignment (Optional)

Image alignment is an optional image pre-processing step, yet it is essential if there are significant shifts between phase images, because cell tracking is accomplished by comparing cell locations between consecutive frames. Alignment is not required if the image registration does not shift significantly. If running the alignment on an image with a large number of moving objects, we suggest using an ROI during the alignment (*see* **Note 15**). Align the images as follows (*see* Video 1 for a video tutorial):

1. Answer "y" when asked, "Do you wish to align images?"
2. If using an ROI to base the alignment upon (optional), make an ROI file (*see* Subheading 3.4) that indicates a stationary area of the image (the largest possible background region that contains a minimal number of moving objects throughout the images of the image set; *see* **Note 15**).

Fig. 3 (continued) Python script in the terminal. Note that the directory the script is running from can be different. (**b**) The *freehand* selection tool (purple) is useful for selecting cells for cell segmentation. The *line* tool (red) is useful for outlining cells and dividing cells to add to the "Not cell" label (*see* **Note 22**). (**c**) An example image loaded in *Fiji* for training in *Weka* by going to [Plugins > Segmentation > Trainable *Weka* Segmentation]. To adjust the parameter and to make labels, click on *Settings* (red). The image loaded into *Weka* has magnified to demonstrate cell selection (*see* **Note 13**). (**d**) Left: the settings used to classify the phase images to a probability mask. Right: the settings used to classify the phase images to a binary mask (*see* **Note 20**). To follow our example, we change the *Weka* default names from "Class 1" and "Class 2" to "Cell" and "Not cell" (*see* **Note 21**)

Table 4
Fiji and Weka terminology

Term	Definition
Classifier	A machine learning algorithm that categorizes input data
Binary mask	A black and white image in which pixels identified as a given label by the classifier are black, and the remaining image is white
Probability mask	A grayscale image based on the probability calculated by the classifier that each pixel is a given label

3. If using an ROI, answer "y" when asked, "Do you have an ROI file for a stationary area?" If aligning based on the whole image, answer "n" for the same prompt.
4. If using an ROI, enter the path to the *csv* file, relative to the working directory.
5. Enter the name of the directory into which images will be saved.

3.6 Cell Segmentation

A single-cell analysis is primarily dependent upon cell segmentation; we use the *Trainable Weka Segmentation* tool in *Fiji*. Weka uses machine learning to train a classifier based on training data selected by the user [27] (Table 4). Subheading 3.6, **steps 1** and **2**, covers how to use Weka to classify images, and we have developed our custom scripts to make the process of training and applying a classifier faster and more direct (*see* **Note 16**; *see* Video 1 for a video tutorial).

1. Subheading 3.6, **step 1**, details how to use *Trainable Weka Segmentation* to classify images. We have had success in the past using two rounds of classification because this method improves segmentation of neighboring cells (*see* **Note 17**). Answer "y" when asked, "Do you wish to train and/or apply a classifier?"
2. Input the full path to the *Fiji* executable file located in the Fiji.app directory when prompted (*see* **Note 12**).
3. Enter how many rounds of classification you are running; we recommend two rounds (*see* **Note 17**).
4. If using a *Linux* machine, answer "y" if you would like to classify the images in the background (it is faster). Then enter how many processes are available for multiprocessing. The number of threads available for these processes will depend on the number of processors available to the individual machine (less if using a virtual machine). Do not use more than half of the available threads for multiprocessing.