ImageJ and CellProfiler: Complements in Open-Source Bioimage Analysis

Ellen T.A. Dobson,[1] Beth Cimini,[2] Anna H. Klemm,[3] Carolina Wählby,[3] Anne E. Carpenter,[2,6] and Kevin W Eliceiri[1,4,5,6]

[1]Laboratory for Optical and Computational Instrumentation (LOCI), Center for Quantitative Cell Imaging, University of Wisconsin at Madison, Madison, Wisconsin
[2]Imaging Platform, Broad Institute of Harvard and MIT, Cambridge, Massachusetts
[3]Science for Life Laboratory BioImage Informatics Facility and Department of Information Technology, Uppsala University, Uppsala, Sweden
[4]Department of Medical Physics, University of Wisconsin at Madison, Madison, Wisconsin
[5]Morgridge Institute for Research, Madison, Wisconsin
[6]Corresponding authors: *anne@broadinstitute.org*; *eliceiri@wisc.edu*

ImageJ and CellProfiler have long been leading open-source platforms in the field of bioimage analysis. ImageJ's traditional strength is in single-image processing and investigation, while CellProfiler is designed for building large-scale, modular analysis pipelines. Although many image analysis problems can be well solved with one or the other, using these two platforms together in a single workflow can be powerful. Here, we share two pipelines demonstrating mechanisms for productively and conveniently integrating ImageJ and CellProfiler for (1) studying cell morphology and migration via tracking, and (2) advanced stitching techniques for handling large, tiled image sets to improve segmentation. No single platform can provide all the key and most efficient functionality needed for all studies. While both programs can be and are often used separately, these pipelines demonstrate the benefits of using them together for image analysis workflows. ImageJ and CellProfiler are both committed to interoperability between their platforms, with ongoing development to improve how both are leveraged from the other. © 2021 Wiley Periodicals LLC.

Basic Protocol 1: Studying cell morphology and cell migration in time-lapse datasets using TrackMate (Fiji) and CellProfiler
Basic Protocol 2: Creating whole plate montages to easily assess adaptability of segmentation parameters

Keywords: cell tracking • CellProfiler • image stitching • ImageJ • segmentation

How to cite this article:
Dobson, E. T., Cimini, B., Klemm, A. H., Wählby, C., Carpenter, A. E., & Eliceiri, K. W (2021). ImageJ and CellProfiler: Complements in open-source bioimage analysis. *Current Protocols*, *1,* e89. doi: 10.1002/cpz1.89

INTRODUCTION

ImageJ and CellProfiler are two of the most widely used open-source image analysis platforms in science. Over the years, both projects have developed independently, cementing themselves in the field of image analysis. Cited in over 10,000 publications per year, ImageJ, and its plugins-included distribution known as Fiji, enable in-depth image investigations; both are ideally suited for interactive, single-image processing and are popular choices for both novice and experienced developers alike to add functionality (Rueden

CURRENT PROTOCOLS
A Wiley Brand
Current Protocols e89, Volume 1
Published in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpz1.89

et al., 2017; Schindelin et al., 2012; Schneider, Rasband, & Eliceiri, 2012). With over 10,000 publications to date, CellProfiler's strength is in project building, where complex pipelines of individual modules can be built to automate the processing of large experiments of images (Carpenter et al., 2006; Kamentsky et al., 2011; McQuin et al., 2018). The projects also differ in their respective languages, with ImageJ being Java based and CellProfiler Python based.

The ImageJ and CellProfiler teams have worked together for many years, not just on mutually useful software development, but also in developing the bioimage analysis community into its present healthy and cooperative state. For example, the teams advocate together for open-source technologies, usability, and interoperability (Carpenter, Kamentsky, & Eliceiri, 2012; Kamentsky et al., 2011). The two software projects joined forces to build the Scientific Community Image Forum (*https://forum.image.sc/*; Rueden et al., 2019), an online resource for members of the worldwide scientific community to learn about image analysis from each other. Currently, 40 open-source imaging software projects, termed "Community Partners," use the Forum as their primary interaction with users. This Forum has increased communication among developers of various software packages and provided users access to a wide breadth of experts in various open-source platforms. The Forum has blossomed into a solid community, with over 14,000 users worldwide, encouraging open science and reproducible research by advocating for open tools and their interoperability.

Recently, the ImageJ and CellProfiler teams at the University of Wisconsin–Madison and the Broad Institute have formally come together to pursue their common open-source development and outreach goals by creating the Center for Open Bioimage Analysis (COBA; *https://openbioimageanalysis.org/*) via a P41 grant from the National Institutes of Health National Institute of General Medical Sciences (NIH NIGMS). COBA's mission is to serve the cell biology community by catalyzing the entire open-source bioimage analysis software field to develop and implement cutting-edge methods, such as deep learning, making them more easily accessible. No single piece of software can handle the full diversity of bioimaging needs. Making tools interoperable and demonstrating how to link them together is therefore a priority for COBA.

Here, we aim to highlight the strengths and weaknesses of ImageJ and CellProfiler and showcase how these programs can be used in conjunction, and therefore more powerfully, to build effective analysis workflows. We also give two use cases that illustrate the complementary strengths of the two programs. The first protocol (Basic Protocol 1) involves integrating the Fiji plugin TrackMate, for semi-automated cell tracking, into a CellProfiler pipeline for further analysis and measurements of cell morphology and migration in time-lapse datasets. The second protocol (Basic Protocol 2) involves using Fiji's fast image manipulation and image exploration tools, specifically the Hyperstack and Make Montage tools, to help verify that the segmentation results of a CellProfiler pipeline are robust across a number of different image phenotypes. In both cases, the strengths of each open-source platform are employed in a single workflow that best addresses researchers' varying needs. We also describe future plans to improve upon these two projects' interoperability. Overall, ImageJ and CellProfiler plan to continue their development as separate programs, while constructively and collaboratively improving researchers' experiences and accessibility to leading image analysis techniques.

BASIC PROTOCOL 1

STUDYING CELL MORPHOLOGY AND CELL MIGRATION IN TIME-LAPSE DATASETS USING TrackMate (FIJI) AND CellProfiler

This protocol uses Fiji and CellProfiler to track the movement of cells and additionally analyze the shape of the cells at each time point. Using this protocol, it is possible to

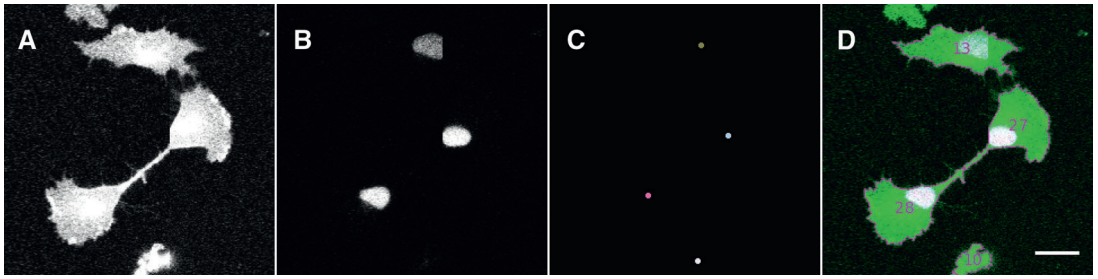

Figure 1 Basic Protocol 1 uses 2-channel time-lapse videos as input: one channel contains a cytoplasmic marker (**A**) and the other channel labeled nuclei (**B**). TrackMate finds positions of the nuclei (**C**) and tracks them over time, while CellProfiler can take the TrackMate positions as input and identify cell outlines (**D**); scale bar, 25 µm.

study the connection between variations in cell morphology and migration over time. The protocol consists of two main steps: tracking the cells using the TrackMate plugin in Fiji and then analyzing cell morphology using CellProfiler. With TrackMate, it is possible to track the cells and also easily visually inspect and correct any errors in the tracking results (Tinevez et al., 2017). CellProfiler is powerful for detecting and quantifying cell morphology.

The workflow uses 2-channel time-lapse videos as input: one channel contains a cytoplasmic marker (Fig. 1A), the other channel labeled nuclei (Fig. 1B). The positions of the nuclei over time are tracked using TrackMate. TrackMate returns the center of the tracked object (visualized as small dots in Fig. 1C), but no information on the size and shape of a tracked object. Using the module "IdentifySecondaryObjects" in CellProfiler, we can use the position of the tracked nuclei as seeds for a 'seeded' watershed, which identifies the outline of the cells by starting at the seed positions and following the intensity profiles of the cytoplasmic marker image. We can then automatically extract a large number of size and shape measurements using CellProfiler and link them back to the tracking results from TrackMate.

Materials

CellProfiler 4.0.7+ (cloned from *https://github.com/CellProfiler/CellProfiler* or downloaded from *https://cellprofiler.org/releases*)
Current version of Fiji (downloaded from *https://imagej.net/Fiji.html#Downloads*)
The following files are all freely available via Zenodo (*https://zenodo.org/record/4317505#.X9OvOflKhPY*):
Fiji IJMacro script: `create_LabelledMasks.ijm`
CellProfiler pipeline: `TrackMate_CellProfiler.cppipe`
Matlab script: `Plot_per_cell.m`
Saved, manually curated TrackMate project: `crop_1_60_ManualCuration.xml`
Saved Spots Results Table of manually-curated TrackMate project: `Spots in tracks statistics.csv`
CellProfiler pipeline output: `cp_output.zip`
Example dataset: `crop_1_60.tif` (a subset of image data from Shafqat-Abbasi et al., 2016)
Stack of labeled masks, output from TrackMate: `crop_1_60_TrackMate_LabelledMask.tif`

Using TrackMate in Fiji: Preparations

1. Load your data (for this example, `crop_1_60.tif`) into Fiji by drag and drop or using "File -> Open."

For tracking, we only need a time-lapse series of the fluorescently labeled nuclei. However, it is easier to inspect the tracking results in a hyperstack that contains both channels (nuclei and cytoplasmic marker) over time. If the two markers are saved into two separate files, open both files in Fiji and combine them by running "Image -> Color -> Merge Channels."

Track cell nuclei using TrackMate

2. Start TrackMate: "Plugins -> Tracking -> TrackMate."
3. Double-check the calibration settings (Pixel width, Pixel height, Time interval), and correct if necessary. Click Next.
4. Select the LoG detector.

   The LoG detector applies a Laplacian of Gaussian (LoG) filter before detecting local intensity maxima. The detector works best for bright round objects. Nuclei are not perfectly round; however, the LoG detector still tracks nuclei well in many cases. Errors in the detection, and therefore the tracking, can be expected in cell lines with many non-round nuclei. Using CellProfiler's IdentifyPrimaryObjects module to identify and export identified nuclei may produce better results in such cases. See CellProfiler's Help for that module for details.

5. In the GUI interface, select the channel with nuclei for object segmentation. Enter an estimated blob diameter and a threshold. Click Preview for inspecting the settings. Settings for the example dataset: segmented channel = channel 1, estimated blob diameter = 20; threshold = 0.5.

   The estimated blob diameter should reflect the expected nuclei diameter. If the entered estimated blob diameter is set too low, the detector will find more than one detection per nucleus. If the blob diameter is too high, difficulties can arise when two cells are close to each other. You can estimate the diameter of a representative nucleus using the line tool in the main toolbar of Fiji. Setting the threshold is a balance between detecting background signals (threshold too low) and missing nuclei (threshold too high).

6. Wait until nuclei are detected. Click Next.
7. Initial thresholding; Click Next.

   The initial thresholding can be adapted; however, in most cases, this is not needed when the threshold in the LoG detector step was chosen properly.

8. Wait for the HyperStack Displayer to load. Click Next.
9. Set filters on spots. For the example data, it is not necessary to add a filter here. Click Next.

   Adding filters can help if there are too many erroneous spots detected. For adding a filter, press the green "plus" symbol, and select a measure parameter to filter on in the drop-down menu.

10. Select a tracker. Select the Simple LAP tracker.

    The Simple LAP tracker does not account for cell division.

11. When choosing the Simple LAP tracker, the following parameters have to be set: Linking max distance, Gap-closing max distance, Gap-closing max frame gap. Settings for the example dataset: linking max distance = 25 microns, gap-closing max distance = 25 microns, gap-closing max frame gap = 2. Click Next.

    Linking max distance is the maximum distance the nuclei are expected to move from frame to frame. To define a good value, draw a line using the line tool in the main toolbar of Fiji and move one end of the line onto the center of one nucleus. Activate the next frame

and drag the other end of the line to the new position of the nucleus. Then, measure the length of the line using "Analyze -> Measure."

Tracks can have gaps, e.g., when cells leave the field-of-view and then return or when the detection fails in a frame. Gap-closing max distance is the maximum distance an object can move within a gap to still be considered belonging to the same track. Gap-closing max frame gap is the number of frames allowed for a gap.

12. Wait for the Tracker to be executed. Click Next.

13. Set filters on tracks. If needed, add filters by pressing the green "plus" symbol. For the example data, no filters were used.

    The choice of track filters depends on the dataset. For example, in a dataset where all cells move, using Track displacement might help to filter out bright particles that stick to the surface and are detected. Number of gaps can be useful to filter out tracks of low detection quality. Be careful to not bias the data by filtering out tracks.

Save the analysis and export the tracking result

14. Click Save to the analysis as an `.xml` file.

    It is important to save the `.xml` file for documentation of all tracking parameters. After saving, click Resume.

15. Click Analysis. Save the table "Spots in tracks statistics" as a `.csv` file using "File -> Save As…".

    Automatically, TrackMate will open three results windows: one window for measurements of detected nuclei (spots), one window for the links from spot to spot, and one window for entire tracks. For further analysis in this protocol, only the file `Spots in tracks statistics.csv` is needed. Please see Critical Parameters below for how to tune and manually correct tracking results in case the output from the automated tracking is not satisfactory. Only a few manual corrections had to be done in the example dataset using the parameters as specified in the protocol. The corrections consisted of closing gaps, adding undetected frames at the end of a track, and splitting dividing cells. Load the corrected `.xml` file, `crop_1_60_ManualCuration.xml`, and inspect the corrected tracking result for the example dataset. The corresponding results table listing the spots of the manually corrected data is available as `Spots in tracks statistics.csv`.

Run create_LabelledMasks.ijm

he TrackMate results table `Spots in tracks statistics.csv` contains a list of all spots (here nuclei detections) and gives their *x*,*y* position at specific time points, *t*. It also lists to which track spots belong by listing the Track ID. The script, `create_LabelledMasks.ijm`, creates a new time-lapse stack where for each time point. The detections are represented as dots with a gray value encoding the TrackID. The script is written in the IJ macro language.

16. Open the time-lapse file used for tracking and the "Spots in tracks" results table output from TrackMate.

17. Open `create_labelledMasks.ijm` in Fiji by drag and drop. The script opens within the Script Editor. Click Run.

18. Input dialog window. The macro needs to know the names of the time-lapse file and the "Spots in tracks" results table. Adjust the defaults, if needed. Click OK.

19. Select the directory where the macro will save the output. Click Select.

20. The macro saves a stack of labeled masks into the chosen output directory. The name of this file is `NameOfTimeLapseFile_TrackMate_LabelledMask.tif`.

The labeled mask corresponding to the corrected tracking results of the example dataset is available online (https://zenodo.org/record/4317505#.X9OvOflKhPY).

Run CellProfiler

21. Open CellProfiler and import the pipeline `TrackMate_CellProfiler.cppipe` (available in the same Zenodo folder as the input data) by dragging it to the part of the screen that says "Drop a pipeline file here" or via "File -> Import -> Pipeline from File…".

22. To save the output of the pipeline after running, set the Default Output Folder using "Output Settings -> Default Output Folder."

23. *Module Images:* Drag and drop the original time-lapse file and the stack of labeled masks that were output from TrackMate into the files list. In the example dataset, these files are named `crop_1_60.tif` and `crop_1_60_TrackMate_LabelledMask.tif`.

24. *Module Metadata:* Click "Extract metadata" and "Update." Check the listed files; each row represents a frame of the original time-lapse movie or a labeled mask from TrackMate.

    For the example dataset, there will be 180 rows in total. `crop_1_60.tif` consists of a total 120 rows representing the 60 time points, and for each time point "T," there are two channels "C" (0, 1). `crop_1_60_TrackMate_LabelledMask` will be listed as 60 rows, reflecting each time point of the stack, since the LabelledMask has only one channel.

25. *Module NamesAndTypes:* Click "Update." For each time point there should be three types of images, labeled as "cytoplasm_marker," "nuclei_marker," and "trackMate_result." The pipeline assumes that the nuclei are in channel 1 and the cytoplasm marker is in channel 2. If you have a reversed order, change to "Metadata Does Have C matching 1" for the "nuclei_marker" images and "Metadata Does Have C matching 0" for the "cytoplasm_marker" images.

26. Choose "Windows -> Hide All Windows on Run."

27. Run "Analyze Images."

28. Check the output in the Default Output Folder.

    The output consists of a control image for each time point which shows the nuclei marker (magenta), the cell marker (green), the outlines of the cells as detected by CellProfiler (magenta), and the TrackID. Note that the CellProfiler TrackID = TrackMate TrackID+1. The pipeline also saves four `.csv` files: `cells.csv`, `Experiment.csv`, `Image.csv`, and `trackMate_detection.csv`, with `cells.csv` containing all the important information for this workflow. `cells.csv` lists many parameters quantifying the size and shape of the cells, as exported by the MeasureSizeShape module of CellProfiler. Read more about the details of the measurement in the module's Help. It also contains the measurement "Intensity_MaxIntensity_trackMate_result_rescaled," which reflects the CellProfiler TrackID (= TrackMate TrackID+1).

BASIC PROTOCOL 2

CREATING WHOLE-PLATE MONTAGES TO EASILY ASSESS ADAPTABILITY OF SEGMENTATION PARAMETERS

CellProfiler is highly adaptable; researchers can create image analysis pipelines for large datasets and tune segmentation parameters to perform well across many different perturbations and phenotypes. There is a major challenge, however, for datasets containing large numbers of image phenotypes, it can be extremely time consuming to check segmentation parameters in enough fields to ensure that they reliably segment images of all phenotypes present in the set. This poses a particular challenge when some

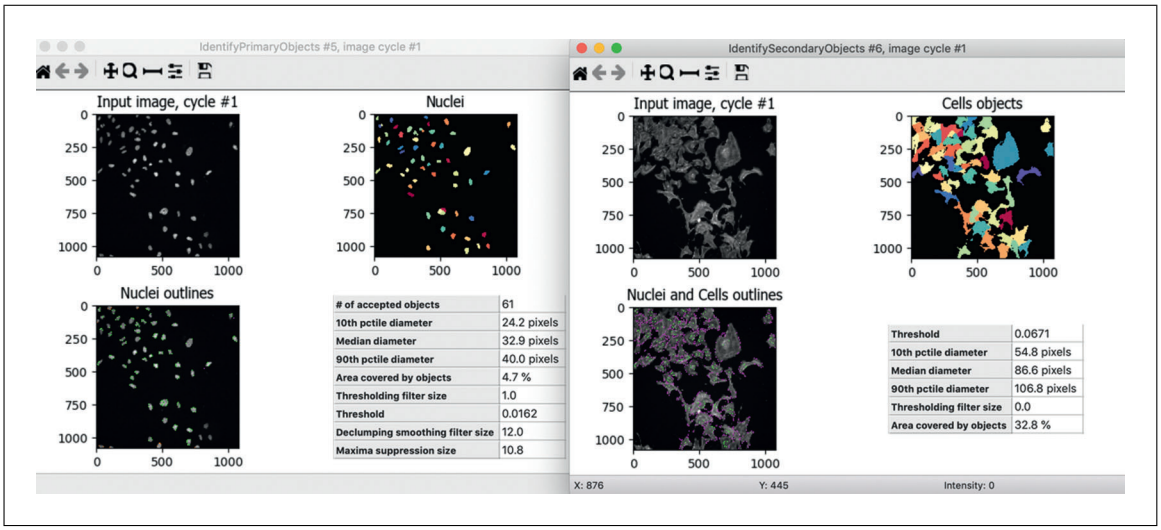

Figure 2 An example for Basic Protocol 2 of primary (left) and secondary (right) object segmentations of a particular site in CellProfiler. These images show reasonable segmentation. Image axes are pixels. Image resolution is 0.656 µm/pixel.

phenotypes are rare, and the screener does not know which treatment(s) are likely to produce these rare phenotypes. Missing a rare phenotype at the segmentation-checking-stage can lead to poor segmentation of this phenotype, potentially causing at least one of several undesirable outcomes: (a) a treatment that did produce a real change, but the change is not captured in the segmentation and a true screening "hit" is identified as a "miss," (b) a poor segmentation that leads to the screener thinking a "hit" changes one aspect of the cell (size) when it truly changes another (intensity), causing the screeners to incorrectly pursue the wrong mechanism of action, and (c) a screener only later realizing that their "hits" are due to segmentation errors and needing to rerun dozens or hundreds of plates worth of segmentation and measurement results, costing both human and computer time. All of these are worth avoiding where possible, but in practice having a screener hand-check tens of thousands of segmentation results one-by-one is tedious, and doing it quickly may cause screeners to miss the rare phenotypes for which they were so diligently searching.

It is possible to use a workflow that allows easy checking of segmentations *en masse*. It begins by running a CellProfiler pipeline that, after manually selecting segmentation parameters for a subset of sample images (Fig. 2), is designed to export one segmentation image per well of a multiwell plate, followed by a Fiji script that automatically stitches these into a "pseudo-plate" image for easy review using Fiji's Hyperstack and Make Montage tools (Fig. 3). While this does not guarantee finding *all* rare phenotypes, since the screener is only visualizing the results of one field per well (and typically more than one field is acquired in each well), it creates a more acceptable tradeoff between the hands-on time required by the image analysis specialist and the likelihood that most rare phenotypes will be viewed and have their segmentation assessed. In theory, a CellProfiler-only version of this workflow can be created using the CellProfiler Tile module, but the CellProfiler-Fiji hybrid solution is orders of magnitude faster to execute and higher quality, as it allows more customized viewing of the output plate montages in the end; in Fiji, brightness and contrast can be adjusted for the whole plate, individual channels can be toggled on and off, the lookup tables of individual channels can be adjusted, and so on.

The CellProfiler section of the protocol here assumes that the dataset contains two channels, one a nuclear marker and one a whole-cell marker, but this approach, and the downstream Fiji script, are adaptable to any number of channels and/or objects; information on how to customize the pipeline to your exact needs is provided in the notes.

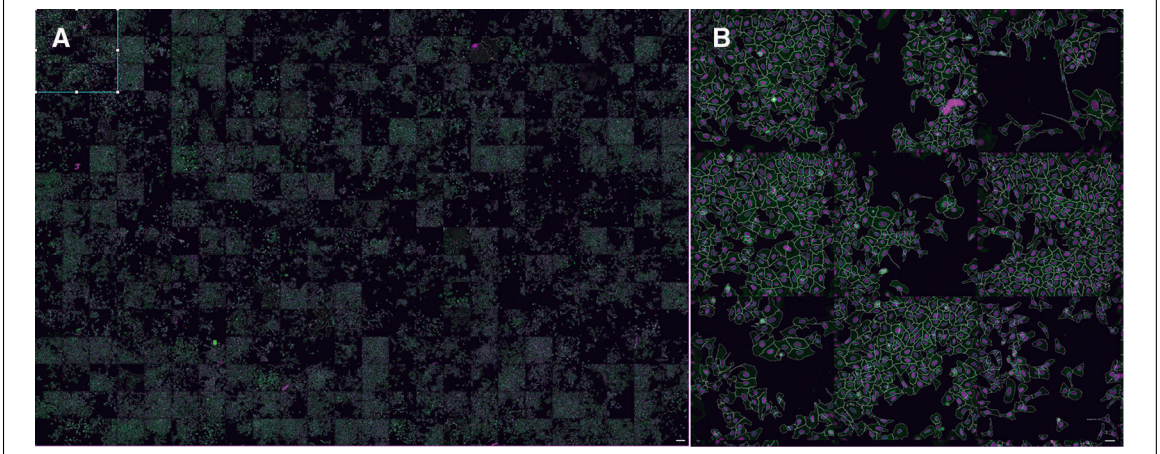

Figure 3 For Basic Protocol 2, a whole-plate stitch (**A**) with area to be inset selected; scale bar, 250 µm. Inset of the whole plate stitch covering wells A01-C03 (**B**); scale bar, 50 µm. In both images, magenta = DAPI, green = phalloidin + WGA, cyan = nuclear outlines, and white = cell outlines.

Materials

CellProfiler 4.0.7+ (cloned from *https://github.com/CellProfiler/CellProfiler* or downloaded from *https://cellprofiler.org/releases*)
Current version of Fiji (downloaded from *https://imagej.net/Fiji.html#Downloads*)
`BBBC025_AssayDevelopment.cppipe` (provided as Supporting Information)
`CellCounts.ipynb` (provided as Supporting Information)
`Make_fiji_montages.py` (provided as Supporting Information)
Test image dataset, BBBC025 (*https://bbbc.broadinstitute.org/BBBC025*)

Configure image sets in CellProfiler

1. Open CellProfiler and import the CellProfiler `.cppipe` pipeline file by dragging to the part of the screen that says "Drop a pipeline file here."
2. Load images by dragging a folder of images to the part of the screen that says "Drop files and folders here."
3. Configure the Metadata module to extract metadata from the file and/or folder names; at a minimum, information about the plate name, well name, and site should be extracted for each image.

   This information is required to make sure that the pipeline processes a single site for each well and that it correctly outputs the images into a separate folder for each plate.

4. Configure the NamesAndTypes module to recognize the channels in your image set.

   NamesAndTypes is currently set up for two input channels (one corresponding to nuclei, and one corresponding to cells), but to customize to your own data you can use anywhere between one and hundreds of channels; see the module help for more information. NamesAndTypes is currently set up to use Site 1 from each well; if you would prefer a different site, you can change the "Metadata Does Have Site matching 1" line for each channel to use whichever site you like.

Perform initial assay development in CellProfiler

5. Open CellProfiler's Test Mode by clicking the "Start test mode" button at the bottom left corner of the program.

   CellProfiler can be run in two modes: TestMode, where one image is assessed at a time, step-by-step, interactively, and AnalysisMode, where all images are run sequentially.

While you are testing your settings, you will want to use TestMode; once your configuration is complete, you will use AnalysisMode.

6. Hit the "step" button twice to run the IdentifyPrimaryObjects and IdentifySecondaryObjects modules on your initial dataset to visualize your initial nucleus and cell segmentation, respectively.

   CellProfiler can be set up to analyze as many types of objects as you would like; in general, add more IdentifyPrimaryObjects modules to identify objects purely from a given image channel, and IdentifySecondaryObjects modules to identify objects based on a given channel plus a "seed object." For example, one could identify nuclei in the DNA stain channel using IdentifyPrimaryObjects, then use IdentifySecondaryObjects to expand those nuclei to find cell borders in a channel where cell bodies are stained. See the in-program Help for each module (buttons marked with "?"), as well as Critical Parameters in the Commentary of this article for links to more resources. If you do add/subtract/change objects in the pipeline, you will need to perform the optional step 11 in this protocol.

7. If needed, adjust the parameters in IdentifyPrimaryObjects and/or IdentifySecondaryObjects to improve segmentation quality on your initial image set. Click "step" to execute any changes you make.

   The "?" button next to each setting gives information on what the setting does, as well as advice on adjusting it. See Critical Parameters for links to more comprehensive resources on adjusting these modules.

8. Choose another image at random by going to the "Test" menu and selecting "Choose random image set."

   If, instead of selecting an image set at random, you wish to navigate to a particular image, you may do so by going to the "Test" menu and then selecting "Choose Image Set." A list of all of the loaded images and their metadata values will be populated. You can sort by whichever metadata value you like; select the image you want to test and then click "Ok" to load that image into TestMode.

9. Repeat steps 6-8 until your segmentation parameters look suitable on many consecutive image sets. This may take many iterations. Reasonable segmentation examples are shown in Figure 2.

   Save your work intermittently by going to File -> SaveProject to avoid losing your work if a crash should occur.

Prepare CellProfiler to run assay development on plate(s) of data

10. Exit Test Mode by clicking the "Exit Test Mode" button.

11. *Optional:* If needed, adjust your OverlayOutlines and SaveImages settings.

    This is necessary if you have adjusted the number of channels, names of channels, number of objects, or names of objects; generally, you need one OverlayOutlines module per object, and one SaveImages module for each image and object to be assessed. See Troubleshooting for more information if you have trouble with this step.

12. Prevent the opening of any windows during the final run by selecting the "Windows" menu, and then selecting the option "Hide All Windows On Run."

    This reduces processing time; you can keep individual windows open to monitor if you prefer.

13. Select the "Output Settings" button and set the Default Output Folder where you want the output image files to be saved.

    You do not need to adjust the Default Input Folder in this pipeline. The Images module contains the information about image file location, and there are no other inputs.

14. Save the final copy of your project file, containing the image and pipeline information, by going to File -> Save Project.

15. Click "Analyze Images" to analyze all images.

Create plate montages

16. Open Fiji, and open the Fiji script, `make_fiji_montages.py` (provided as Supporting Information), by dragging-and-dropping the script onto the main Fiji toolbar.

    Alternatively, to make this script permanently accessible from within Fiji, rename the script such that it starts with an underscore ("_") and put it in your Fiji installation's "Plugins" folder; upon rebooting Fiji, the script can be accessed in Fiji's "Plugins" menu for as long as the file is in this location.

17. Execute the script by clicking "Run."

    If you are executing this script from the plugins menu, simply click on the name of the plugin in the plugins menu.

18. Answer the script's prompts to provide the folder where your CellProfiler-produced images were saved (i.e., the CellProfiler output folder that contains the folder of output images and the `.csv` files), as well as the number of columns and rows, and the scale (1× size the original images, 0.5×, etc) at which you want to save the output montage.

19. Allow the script to execute. "Command finished" will appear in the Fiji bar when it has completed.

Review plate montage images

20. When processing completes, you may open the finished montages in Fiji.

21. Examine the montages (Fig. 3) for all plates to check for appropriate segmentation in all wells. Note any plates and/or wells you think are performing particularly poorly.

    To adjust the zoom of your montage, use the "Image -> Zoom" menu; to toggle colors on and off, use the "Image -> Color -> ChannelsTool" menu-item-tool; to adjust the brightness of each channel, use the "Image -> Adjust -> Color Balance" menu item tool; to adjust the displayed color of each channel, make sure that channel is selected in Fiji using the channel slider at the bottom of the montage, and then select a new colormap in the "Image -> Lookup Tables" menu.

22. When all plates have been examined, assess if you think the segmentation errors you have seen warrant adjustment of your segmentation pipeline. If so, repeat steps 5-21 as needed, using the notes you made in step 21 to guide your manual image selections as you run step 8.

    Be sure to check some images that worked well (plates and wells that were not on your list), as well as ones that worked poorly, to make sure you are not sacrificing segmentation of most images to fix things for only few, rare phenotypes. Achieving perfect segmentation is typically impossible, so your goal is simply to maximize the proportion of samples with suitably decent segmentation results.

23. When you are satisfied with your segmentation parameters, you may finalize your CellProfiler pipeline to prepare it for full-scale analysis by (a) removing the site filter in your NamesAndTypes module, (b) removing all modules after the initial segmentation modules, (c) adding any measurements you need for your phenotypes, and (d) saving your project with a new name using the "File -> Save Project As" function in CellProfiler.

Sample data

The sample pipeline and script are designed to be run on image set BBBC025 (*https://bbbc.broadinstitute.org/BBBC025*) from the Broad Bioimage Benchmark Collection (Ljosa, Sokolnicki, & Carpenter, 2012). The pipeline minimally requires the Hoechst and PhGolgi channels from any single plate, but was optimized on plates 37,983 + 38,002; the additional channels of this experiment can easily be added as described in the CellProfiler sections of the protocol.

COMMENTARY

Background Information

ImageJ has been a mainstay in the image analysis community since its inception in 1987 at the National Institutes of Health (NIH). Originally called NIH Image, it was developed by Wayne Rasband with the intention to be easily and openly shared and extended via code contributed by the scientific community at large. Rasband proposed from the start the idea that this program would have third-party contributors that could customize and develop their own analysis components, making this a user-driven open-source project. Sun Microsystems released a Java programming language that could run on any operating system in 1995, which would allow programmers to "write once, run anywhere." This is what led to the transition of NIH Image, originally written in Pascal, to Java, and to the first release of ImageJ (Schneider et al., 2012).

The driving design concept behind NIH Image and ImageJ was to keep things simple, in particular the interface, which is minimalist and has remained essentially the same to this day. However, this interface 'hides' the true power and extensibility behind this platform. Because of the developer community orientation of ImageJ, it had to be made easily extensible by outsiders, as opposed to a centralized development model. Rasband achieved this through the use of macros and plugins. Macros are short, customizable scripts that allow researchers to automate their analysis workflows using existing features and commands within the software. Plugins are more advanced, modular software elements that extend the functionality of the software. Plugins were either integrated into ImageJ if Rasband saw their global utility or were available for download from third-party sites (Schneider et al., 2012). This decentralized development is what has led to one of ImageJ's greatest strengths as an image analysis platform: the ImageJ community of users and developers.

ImageJ has, to this day, remained a community-driven open-source project. The development of Fiji, a distribution of ImageJ primarily focused on biological-image analysis which began in 2007, allows for the formal sharing of new algorithms with end users through an integrated update system (Schindelin et al., 2012). ImageJ and Fiji have continued down the original path set out by Rasband years ago: to remain a 'federation' of decentralized developers driven by community needs. The core ImageJ team, which includes developer leadership by the Eliceiri laboratory at the University of Wisconsin-Madison and many collaborators around the world, does develop and improve the architecture of the entire ImageJ ecosystem (Rueden et al., 2017). However, its main goal is to drive independent learning and development as opposed to limited analysis support and code production in-house.

ImageJ was developed with the self-taught coder at the bench in mind. It is oriented towards single-image processing and is quite customizable via scripting, etc. Users can add their own functionality as well, focusing on their own research priorities for analysis solutions. Therefore, the target audience for ImageJ remains the bench biologist who is able to find existing plugins or macros for a given task, and even someone who is willing to put in the time and effort needed to script and expand the functionality of the program. It also targets users who need a platform for initial image viewing, editing, and processing. ImageJ is an ideal platform for efficient image inspection and testing/building potential analysis workflows. Large-scale analysis is possible with ImageJ via integrations with KNIME, an analytics platform for visually constructing nonlinear workflow graphs whose nodes perform steps such as data mining, machine learning, image processing, and text analysis (Berthold et al., 2008; Dietz & Berthold, 2016; Dietz et al., 2020; Fillbrunn et al., 2017).

CellProfiler, another mainstay platform in open-source image analysis, was started in 2003 by Anne E. Carpenter and Thouis (Ray) Jones in the Sabatini Laboratory at the Whitehead Institute for Biomedical

Research and Golland laboratory at MIT's Computer Science & Artificial Intelligence Lab. CellProfiler was originally born out of necessity, as driven by the work of Carpenter, a cell biologist who saw a need for user-friendly software for more advanced, large-scale image analysis (Carpenter, 2020). CellProfiler was initially released in December 2005, originally developed in MATLAB (Carpenter et al., 2006); it was re-written in Python and released as CellProfiler 2.0 in 2010 (Kamentsky et al., 2011). Support for volumetric analysis of 3D image stacks and optional deep-learning modules was added in the release of version 3.0 in October 2017 (McQuin et al., 2018). The recent release of version 4.0 in September 2020 included improvements to speed, usability, and utility, including a migration to Python 3 (*https://carpenterlab.broadinstitute.org/blog/cellprofiler-40-release-improvements-speed-utility-and-usability*). The CellProfiler project is currently actively maintained by the Carpenter lab at the Broad Institute of Harvard and MIT.

Unlike ImageJ, CellProfiler was specifically designed for larger-scale experiments and building modular, reproducible analysis pipelines. The clear inputs and outputs of its modules can be easily pipelined into extensive analysis workflows. Most of CellProfiler's user base is unfamiliar with code, mainly consisting of biologists who aim for user-friendly, "point and click" solutions to their analysis needs. Users rely on the CellProfiler team at the Carpenter lab for their analysis and development expertise; their top-down design results in heavy curation of CellProfiler features and extensive, consistent documentation that readily orients beginners to making their own analysis pipelines without a need for coding.

As both tools encompass a large fraction of the commonly required algorithms required for most image analysis tasks, the choice of ImageJ versus CellProfiler for a given analysis task is often based on the user's personal preferences. ImageJ's interactive nature and ease of image exploration make it a common first tool for exploring any image set, particularly multi-dimensional images (time-lapse, 3D, or both). Users often choose to use ImageJ for an analysis task when taking advantage of the large number of plugins added to ImageJ by its community, when the number of images to be analyzed is small, when the analysis of each image needs to be individually customized, and/or when the user is comfortable with scripting and so can automate a series of steps using ImageJ's fast processing functionalities. Likewise, users often choose CellProfiler to analyze their experiments based on the ability to easily reproduce results and build standard point-and-click workflows from curated, documented modules with clear inputs and outputs, the large measurement suites that make it easy to deeply analyze each image set and carry out machine learning-based classification, and/or the ease with which analyses can be scaled to large image sets without the need to write macros or scripts. For this reason, it is commonly used in pharmaceutical and academic high-content screening centers.

Some algorithms and strategies, however, are only available in one tool or the other, and a language divide (Java vs. Python) makes it non-trivial to exchange code directly between the programs, leading to demands for interoperability. At the simplest level, users can employ the tools in sequence, as demonstrated in the protocols here; in these cases, images saved to disk form the bridge between the first and second tools in the sequence. Enabling ImageJ and CellProfiler to *directly* interoperate began with a bridge to run ImageJ macros in the context of CellProfiler pipelines in CellProfiler 2.1 (Kamentsky et al., 2011), a functionality that has since been deprecated over the years due to maintenance burden. Because of their now formal commitment as a collaborative organization via COBA, ImageJ and CellProfiler have renewed their dedication to creating and maintaining direct bridges between the two platforms. As a result, a new CellProfiler module was developed in CellProfiler 4.0.8 (RunImageJMacro), which allows users to pass images and variables to an ImageJ macro or script from a CellProfiler pipeline and receive images back from ImageJ that they can use in the rest of their CellProfiler pipeline. This bridge is just the beginning of more advanced interoperability to come, such as the ability to use both tools together in the context of scripting and workflow management tools such as KNIME.

Critical Parameters

Basic Protocol 1

The critical parameters in TrackMate are the initial parameters for spot detection (step 5) and the linking parameters. A good tracking result is essential for downstream analyses, and improvements may be achieved adjusting the parameters for detection and track linking

(steps 6, 8, 10, and 14). Read more about the parameters and settings of TrackMate online (*https://imagej.net/TrackMate*).

It is advisable to adjust the lookup table of both channels (nuclei and cytoplasm) for best possible visual inspection of tracks. Do this using "Image -> Color -> Channels Tool." Toggle between "Composite" and "Gray Scale" visualization when inspecting the tracking result.

Manual correction should be avoided as much as possible, since it is time consuming and non-reproducible. In many cases, however, some manual corrections of the tracking results are unavoidable. TrackMate allows users to add and delete detections, and modify their position. Also, several tracks can be linked to one, and one track can be split into several individual tracks. Shortcuts support the user doing this efficiently. Read the documentation about how to manually curate tracking results (*https://imagej.net/Manual_editing_of_tracks_using_TrackMate*). When following this protocol for subsequent cell morphology analysis, the following corrections are crucial:

Avoid gaps in tracks: Add a detection manually, and link it to a track to fill the gap.

Dividing cells: The Simple LAP tracker gives one daughter cell the track ID of the mother cell and the other a new track ID. This is useful because for this workflow it is not possible that two cells share the same track ID. Depending on the scope of the analysis, it can make sense to manually split the track after division such that both daughter cells receive a new track ID and are treated individually from the mother cell.

After correcting the tracks, do not forget to save the `.xml` file under a new name, and also click Analysis again to obtain and save a new `Spots in tracks statistics.csv` results file.

The critical parameters in CellProfiler are the parameters of IdentifySecondaryObjects (step 7), since they define the detection of the cell borders.

Image data has to be suitable for tracking and segmentation: the nuclei and cell markers should have a high signal intensity compared to the background, but signals should not be saturated. High cell densities should be avoided, and the frame rate has to be adjusted such that the cells are still easy to link from frame-to-frame (i.e., minimal cell displacement from frame-to-frame).

Basic Protocol 2

Segmentation accuracy is driven by adjusting the suite of settings in IdentifyPrimaryObjects and/or IdentifySecondaryObjects in parallel (steps 6-8). The full description of how to adjust these parameters to any dataset the user may want is outside the scope of this manuscript; the CellProfiler manual (*https://cellprofiler-manual.s3.amazonaws.com/CellProfiler-4.0.7/modules/objectprocessing.html*) contains an explanation of all parameters, and our written (*https://github.com/CellProfiler/tutorials*) and video (*https://www.youtube.com/playlist?list=PLXSm9cHbSZBBy7JkChB32_e3lURUcT3RL*) tutorials also cover these concepts in detail. It is not typically possible to find parameters that work perfectly on every cell in every image; the goal is to find parameters that treat all images fairly and work well for the vast majority of cells in each image. See our blog post (*https://carpenterlab.broadinstitute.org/blog/when-to-say-good-enough*) on assessing image quality and when a pipeline is "good enough."

When running the Fiji script, adjusting the scale parameter in the script is essential to the usefulness of the stitched montages. At too small a scale, the montage may be so downsampled that segmentation quality cannot be accurately assessed; at too large a scale, the montage may be too large for the user to open. It is reasonable to start at the default-provided parameter of 1, and adjust empirically based on the size of your original image, the number of channels, and the size of the objects whose segmentation you wish to assess.

Troubleshooting

General

When troubleshooting integrated ImageJ and CellProfiler protocols such as those presented in this article, the best location to discuss hurdles and gain potential solutions is the Scientific Community Image Forum (*https://forum.image.sc/*). Due to the variability and uniqueness of such integrated protocols, specific questions can be posted on the forum to gain insight from experts within the entire open-source community, including the authors of this article, who regularly frequent the forum.

Basic Protocol 1

When observing cells over time and extracting measurements, there are typically two types of errors: errors in the tracks and

errors in how the cells are segmented or outlined. Here, we describe how to detect errors, and what effect they may have on the final result. There are two straightforward ways to detect both segmentation and tracking errors: (1) look for outliers in the final result and (2) browse through and visually inspect the final output. The output from CellProfiler, including outlines of cytoplasms and TrackID numbers from the tracking, can be easily re-opened and interactively viewed in Fiji, as described below. If outliers are detected in the final results, the same approach can be used, focusing on the cells and time frames corresponding to the outliers. The following troubleshooting steps may be appended to Basic Protocol 2:

29. Run "File -> Import -> Image Sequence…" and select the first image of the series in the output folder. Fiji loads the output images into one stack.
30. Browse through the images and inspect whether outlines of cytoplasms have been sufficiently well detected. If there are too many incorrect outlines, go back to CellProfiler and adjust the settings for identifying the cytoplasm in IdentifySecondaryObjects.

For detecting tracking errors, focus on the TrackID number while browsing through the stack. If the number disappears in a frame, the track contains a gap. There is a problem in linking the nuclei if the number changes from one frame to another or if a number appears on top of another nucleus. Go back to TrackMate in Fiji to adjust settings to try to reduce tracking errors.

Changing the settings of IdentifySecondaryObjects:

31. To visually examine the result on a given time frame, click Start Test Mode and use "Test -> Choose Image Set" in the top menu to choose one of the time frames in which you detected incorrect segmentations. Note that the time frames start with Frame = 0 for the first frame in CellProfiler, but $t = 1$ in Fiji.
32. Click on the grayed-out eye symbol next to IdentifySecondaryObjects to open the eye.
33. Click Run.
34. In the Module IdentifySecondaryObjects, check which settings work best for the test image.
35. Click Step, and inspect the output of IdentifySecondaryObjects.
36. Load other time frames of the stack using "Test -> Choose Image Set" to check whether the settings are generally applicable to the stack. Click Run to run the pipeline on the chosen test image.
37. Exit Test Mode, and re-run the analysis using Analyze Images.

Re-loading the tracking result in TrackMate and correcting for tracking errors:

38. Run "Plugins -> Tracking -> Load a TrackMate file" to open the `.xml` file saved. TrackMate will open the TrackMateGUI already at the step "Display option" and also the time-lapse movie used for tracking.
39. Correct the tracking result as described in this protocol under 'Tune and Manually Correct the Tracking Result'.
40. Save the new `.xml` file under a new name using the "Save" button of the TrackMate GUI.
41. Click Analysis, and save the results table "`Spots in tracks statistics.csv`."
42. Re-run `create_LabelledMasks.ijm` as described under the section 'Run `create_LabelledMasks.ijm`' in this protocol.

Effect of errors

The effect of errors while tracking depends on the goal of the analysis. If the goal is to monitor an overall effect of, e.g., adding a perturbant to the cell culture, mixing up two tracks may have little effect on the final result. However, if the actual behavior of individual cells in a heterogeneous population is crucial, care should be taken to correct the TrackMate output.

Gaps in tracks can result in segmentation errors if the cell containing the tracking-gap is close to another one.

The CellProfiler pipeline may over- or under-estimate the shape, size, and intensity measurements of individual cells as a result of segmentation errors. This may happen if the cytoplasms become very extended and fragmented, meaning that long protrusions may partially fall below the intensity threshold in the segmentation step. To keep track of the potential effect of segmentation errors, it may be a good idea to add a step to the pipeline where the total number and area of cytoplasmic objects that are above the intensity threshold are measured. Comparing this number to the total cell area will give a hint on cell fragmentation in large-scale experiments.

Basic Protocol 2

Users should take care that, when adjusting the number or names of channels used and/or the number or names of objects whose segmentation is to be tested, they adjust any necessary modules downstream of their changes. Modules that need to be adjusted will have a red "X" next to them, which will tell you the error if you hover over it; modules that do not need to be adjusted will have green checks next to them.

As a rule of thumb, the CellProfiler pipeline should include one SaveImages module for every raw image the user wants to see in their segmentation montage and one Identify(Primary/Secondary/Tertiary)Objects + one OverlayOutlines + one SaveImages module for each object whose segmentation needs to be examined. Users may include more modules as they like, such as an ExportToSpreadsheet module to export object counts or measurement modules to assess object properties, such as size or intensity, for troubleshooting. In order to correctly batch the data by plate, it is important that each SaveImages module use the "Default Output Folder SubFolder" option, with the subfolder set to the plate metadata so that the Fiji script has the structure it requires to run.

The Fiji script assumes that all rows and columns are full and that the total number of images present must be divisible by the stated number of rows × columns; it will otherwise refuse to stitch. Users should therefore be careful that all the CellProfiler output data is complete before executing the Fiji script and that no other image files are present in the plate subfolders. No other *subfolders* (i.e., data subfolders) should be present in the folder containing the plate subfolders, but files not in subfolders are fine and will be ignored.

If the actual plate layout is such that not all rows and/or columns contain the same number of wells, the user can choose to load into CellProfiler only those rows/columns that do form a complete rectangle, or can fill in missing data by duplicating and renaming existing image data to stand in for data from missing wells.

Statistical Analysis

General

Data produced by the described protocols can also be exported via Windows Excel, Google Sheets, R, KNIME, etc., for further processing and analysis; use the ExportToSpreadsheet module of CellProfiler (or ExportToDatabase if you prefer that format).

Basic Protocol 1

Statistical analysis depends on the research question. Typical parameters for analyzing cell motility are mean square displacement, analysis of the turning angles between frames, and distribution of turning angles (Beltman, Marée, & de Boer, 2009). CellProfiler outputs a `.csv` file, `cells.csv`, with a large number of measurements per cell and time point. It does not automatically output velocity, but as x and y coordinates are provided for each time frame, this velocity can easily be calculated. Using a short Matlab script, `plot_per_cell.m`, we calculated velocity and created a scatter plot of velocity, form factor, and area, where each dot represents a specific cell at a given time point (Fig. 4A). From this plot, we can see that cells with small area and small form factor have lower velocity, while larger cells tend to have higher form factor and higher velocity. This could be statistically verified by measuring the coefficient of determination (R2) between, e.g., per-cell mean area and mean velocity. Next, we also plotted area per cell over time (Fig. 4B). This plot is useful for determining tracking and cell segmentation quality, as smooth variations over time should be expected. These types of plots are also useful when, e.g., visualizing and quantifying changes over time, such as the effect of a perturbant added at a specific time point. For statistical analysis, mean and standard deviation in per-cell change over time may be useful.

Understanding Results

Basic Protocol 1

As described in the statistical analysis and Figure 4, there are a number of different ways the output data can be used. The `cells.csv` format is a simple comma-separated data file, where each row represents a cell in a given time point. The feature measurements extracted by CellProfiler are named at the top of each column of the data file, and many of them are self-explanatory. For detailed descriptions, see the manual for the corresponding measurement modules in CellProfiler. As mentioned above, it is important to notice that it is the *"Intensity_MaxIntensity_trackMate_result_rescaled"* that reflects the TrackID from TrackMate. The corresponding ID numbers are shown

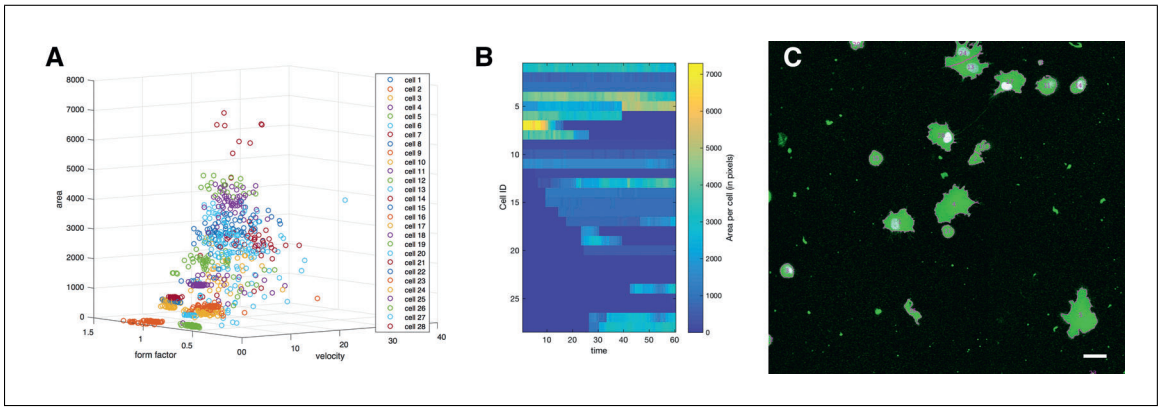

Figure 4 The per-cell and per-time-frame measurements output by CellProfiler for Basic Protocol 1 can be processed in many ways. A scatter plot (**A**) of velocity, form factor, and area, where each dot represents a specific cell at a given time point, shows the relationship between shape and motion, while (**B**) shows variation in area per cell over time. CellProfiler also provides cell IDs and outlines (**C**); scale bar 25 µm.

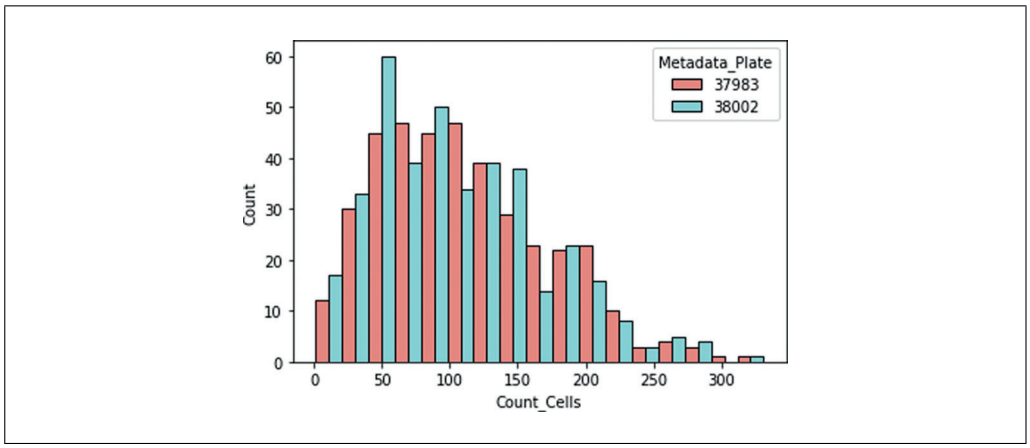

Figure 5 The cell counts from each image exported by CellProfiler's ExportToSpreadsheet module for Basic Protocol 2, as broken down by plate.

on top of the individual cells as shown in Figure 4C.

Basic Protocol 2

It is very difficult to assess segmentation quantitatively without laborious creation of a ground truth; however, in our sample datasets with our preferred segmentation pipeline, we achieved a mean ± standard deviation of 109 ± 62.5 cells per image, distributed as per Figure 5. These data were analyzed in a Jupyter Notebook, `CellCounts.ipynb`, using pandas 1.1.4 and seaborn 0.11.0 (Kluyver et al., 2016; McKinney, 2010; Reback et al., 2020; Waskom & The Seaborn Development Team, 2020), which can also be used to reproduce results of your own segmentation parameters.

Time Considerations

Basic Protocol 1

Processing of the example dataset takes only few seconds (detection and linking) in TrackMate and 2-3 min in CellProfiler (system: Intel^{R} Core^{TM} i7-6700HQ CPU @ 2.60 GHz, 2601 Mhz, 4 Core(s), 8 Logical Processor(s), 32 GB RAM). Manually inspecting the tracking results and correcting them can take up to several hours, and the time investment has to be considered. The decision of whether the time for manually correcting the tracking results is acceptable depends on both the dataset and the research question.

Basic Protocol 2

The amount of time needed to complete the CellProfiler section will depend on the number of images to be run, the number of objects per image, and the number of CPUs available on which to run; the 768 fields of view from two sample plates of BBBC025, which contain on average 100-200 cells per image, took approximately 25 min to complete in CellProfiler 4.0.7 utilizing 4 of the 8 cores of a MacBook Pro (Intel Core i7 3.1 GHz, 16 GB of RAM). A cluster

can be utilized to speed processing of very large datasets; see this manual page (*https://cellprofiler-manual.s3.amazonaws.com/CellProfiler-4.0.7/help/other_batch.html*) and this GitHub page (*https://github.com/CellProfiler/CellProfiler/wiki/Adapting-CellProfiler-to-a-LIMS-environment*) for more information on running CellProfiler on a cluster.

The Fiji section will likewise somewhat depend on the size of the images and the number of channels the user would like to assess; for the sample data, which had a 1080 × 1080 pixel image size, four input channels (two image channels and two outline overlay channels), and 384 wells, execution took 2-3 min per plate on the same machine as the CellProfiler assessment.

Future Directions

There is great value in using the ImageJ and CellProfiler programs separately. However, we hope that we have conveyed the strength in using them together in combined image analysis workflows to best harness their individual strengths. ImageJ and CellProfiler complement each other quite well, and given their commitment via COBA, there remain opportunities to improve existing bridges between them. One of the major hurdles has been the divide in languages for development between Java and Python. However, PyImageJ, a recently developed Python wrapper for ImageJ (*https://github.com/imagej/pyimagej*), allows for the mixing and matching of image processing routines from ImageJ, including plugins and scripts, with Python-side image processing (e.g., scikit-image, ITK, and OpenCV). Therefore, PyImageJ will allow for even more convenient use of ImageJ and CellProfiler platforms in the same workflow, providing a strong step towards improved, long-term interoperability.

Acknowledgments

This work was supported by the National Institutes of Health (NIH P41 GM135019 to A.E.C. and K.W.E.), the SciLifeLab BioImage Informatics Facility, the Swedish Foundation for Strategic Research (SSF–SB16-0046), and Network of European BioImage Analysts (NEUBIAS). We thank the thousands of users and developers of bioimage analysis software who have improved the tools and created a welcoming and helpful community. We also thank Erin Weisbart for her editing of the manuscript, as well as Staffan Strömblad for sharing the example image data used in Basic Protocol 1.

Author Contributions

Ellen T.A. Dobson: Conceptualization, Formal analysis, Investigation, Methodology, Validation, Visualization, Writing-original draft, Writing-review & editing, Beth Cimini: Formal analysis, Investigation, Methodology, Validation, Writing-original draft, Writing-review & editing, Anna H. Klemm: Data curation, Formal analysis, Investigation, Methodology, Validation, Visualization, Writing-original draft, Writing-review & editing, Carolina Wählby: Funding acquisition, Resources, Software, Supervision, Writing-original draft, Writing-review & editing, Anne E. Carpenter: Conceptualization, Funding acquisition, Project administration, Resources, Supervision, Writing-original draft, Writing-review & editing, Kevin W Eliceiri: Conceptualization, Funding acquisition, Project administration, Resources, Software, Supervision, Writing-original draft, Writing-review & editing

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are openly available in *https://zenodo.org/record/4317505#.X9OvOflKhPY https://bbbc.broadinstitute.org/BBBC025* and the supplementary material of this article.

Literature Cited

Beltman, J. B., Marée, A. F. M., & de Boer, R. J. (2009). Analysing immune cell migration. *Nature Reviews. Immunology*, *9*, 789–798. doi: 10.1038/nri2638.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., … Wiswedel, B. (2008). KNIME: The konstanz information miner. In C. Preisach, H. Burkhardt, L. Schmidt-Thieme, & R. Decker (Eds.), *Data analysis, machine learning and applications* (pp. 319–326). Berlin Heidelberg: Springer.

Carpenter, A. E. (2020). Bridging domain and data. *Patterns*, *1*, 100064.

Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I. H., Friman, O., … Sabatini, D. M. (2006). CellProfiler: Image analysis software for identifying and quantifying cell phenotypes. *Genome Biology*, *7*, R100. doi: 10.1186/gb-2006-7-10-r100.

Carpenter, A. E., Kamentsky, L., & Eliceiri, K. W. (2012). A call for bioimaging software usability. *Nature Methods*, *9*, 666–670. doi: 10.1038/nmeth.2073.

Dietz, C., & Berthold, M. R. (2016). KNIME for open-source bioimage analysis: A tutorial. *Advances in Anatomy Embryology and Cell Biology*, *219*, 179–197. doi: 10.1007/978-3-319-28549-8_7.

Dietz, C., Rueden, C. T., Helfrich, S., Dobson, E. T. A., Horn, M., Eglinger, J., … Eliceiri, K. W. (2020). Integration of the ImageJ Ecosystem in KNIME Analytics Platform. *Frontiers in Computer Science*, *2*, 8. doi: 10.3389/fcomp.2020.00008.

Fillbrunn, A., Dietz, C., Pfeuffer, J., Rahn, R., Landrum, G. A., & Berthold, M. R. (2017). KNIME for reproducible cross-domain analysis of life science data. *Journal of Biotechnology*, *261*, 149–156. doi: 10.1016/j.jbiotec.2017.07.028.

Kamentsky, L., Jones, T. R., Fraser, A., Bray, M.-A., Logan, D. J., Madden, K. L., … Carpenter, A. E. (2011). Improved structure, function and compatibility for CellProfiler: Modular high-throughput image analysis software. *Bioinformatics (Oxford, England)*, *27*, 1179–1180. doi: 10.1093/bioinformatics/btr095.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., … Carol and JupyterDevelopment Team (2016). Jupyter Notebooks ? A publishing format for reproducible computational workflows. In F. Loizides & B. Scmidt (Eds.), *Positioning and power in academic publishing: Players, agents and agendas* (pp. 87–90). Amsterdam: IOS Press. Available at: https://eprints.soton.ac.uk/403913/.

Ljosa, V., Sokolnicki, K. L., & Carpenter, A. E. (2012). Annotated high-throughput microscopy image sets for validation. *Nature Methods*, *9*, 637. doi: 10.1038/nmeth.2083.

McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61), Austin, Texas.

McQuin, C., Goodman, A., Chernyshev, V., Kamentsky, L., Cimini, B. A., Karhohs, K. W., … Carpenter, A. E. (2018). CellProfiler 3.0: Next-generation image processing for biology. *PLoS Biology*, *16*, e2005970. doi: 10.1371/journal.pbio.2005970.

Reback, J., Mc Kinney, W., Brockmendel, J., Bossche, J. V., Augspurger, T., Cloud, P., … Pandas Development Team. (2020). pandas-dev/pandas: Pandas 1.1.4. Zenodo. Available at: https://github.com/pandas-dev/pandas/releases/tag/v1.1.4.

Rueden, C. T., Ackerman, J., Arena, E. T., Eglinger, J., Cimini, B. A., Goodman, A., … Eliceiri, K. W. (2019). Scientific Community Image Forum: A discussion forum for scientific image software. *PLoS Biology*, *17*, e3000340. doi: 10.1371/journal.pbio.3000340.

Rueden, C. T., Schindelin, J., Hiner, M. C., DeZonia, B. E., Walter, A. E., Arena, E. T., & Eliceiri, K. W. (2017). ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinformatics*, *18*, 529. doi: 10.1186/s12859-017-1934-z.

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., … Cardona, A. (2012). Fiji: An open-source platform for biological-image analysis. *Nature Methods*, *9*, 676–682. doi: 10.1038/nmeth.2019.

Schneider, C. A., Rasband, W. S., & Eliceiri, K. W. (2012). NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, *9*, 671–675. doi: 10.1038/nmeth.2089.

Shafqat-Abbasi, H., Kowalewski, J. M., Kiss, A., Gong, X., Hernandez-Varas, P., Berge, U., … Strömblad, S. (2016). An analysis toolbox to explore mesenchymal migration heterogeneity reveals adaptive switching between distinct modes. *eLife*, *5*, e11384. doi: 10.7554/eLife.11384.

Tinevez, J.-Y., Perry, N., Schindelin, J., Hoopes, G. M., Reynolds, G. D., Laplantine, E., … Eliceiri, K. W. (2017). TrackMate: An open and extensible platform for single-particle tracking. *Image Processing for Biologists*, *115*, 80–90.

Waskom, M., & The Seaborn Development Team (2020). mwaskom/seaborn. Zenodo. doi: 10.5281/zenodo.592845.

Internet Resources

https://forum.image.sc/

The Scientific Community Image Forum.

https://openbioimageanalysis.org/

The official homepage for The Center for Open Bioimage Analysis (COBA).

https://imagej.net/Welcome

The official wiki homepage for the ImageJ Ecosystem, including ImageJ and Fiji.

https://imagej.net/Fiji/Downloads

The wiki page for downloading Fiji.

https://cellprofiler.org/

The official homepage for CellProfiler.

https://cellprofiler.org/releases

The official page for downloading CellProfiler releases.

https://github.com/CellProfiler/CellProfiler

The GitHub page of CellProfiler for contributing code, maintaining third-party modules, and/or downloading beta releases.

https://github.com/imagej/pyimagej

The code for PyImageJ, a Python wrapper for ImageJ.

https://imagej.net/TrackMate

The official wiki page for the Fiji plugin, TrackMate.

https://imagej.net/Manual_editing_of_tracks_using_TrackMate

A TrackMate tutorial for manual editing of tracks.

https://zenodo.org/record/4317505#.X9OvOflKhPY

Link to all files needed for Basic Protocol 1.

https://bbbc.broadinstitute.org/BBBC025

Link to BBBC025 image dataset for Basic Protocol 2.

https://carpenterlab.broadinstitute.org/blog/cellprofiler-40-release-improvements-speed-utility-and-usability

CellProfiler blog post regarding release of version 4.0.

https://cellprofiler-manual.s3.amazonaws.com/CellProfiler-4.0.7/modules/objectprocessing.html

CellProfiler manual for object processing (adjusting parameters for IdentifyPrimaryObjects and IdentifySecondaryObjects).

https://github.com/CellProfiler/tutorials

CellProfiler tutorial pipelines and images on GitHub.

https://www.youtube.com/playlist?list=PLXSm9cHbSZBBy7JkChB32_e3lURUcT3RL

YouTube page for CellProfiler video tutorials.

https://carpenterlab.broadinstitute.org/blog/when-to-say-good-enough

CellProfiler blog post on assessing image quality and pipeline readiness.

https://cellprofiler-manual.s3.amazonaws.com/CellProfiler-4.0.7/help/other_batch.html

CellProfiler help page for batch processing.

https://github.com/CellProfiler/CellProfiler/wiki/Adapting-CellProfiler-to-a-LIMS-environment

GitHub page showing how to run CellProfiler on a cluster.