Chapter 1

TASBE Image Analytics: A Processing Pipeline for Quantifying Cell Organization from Fluorescent Microscopy

Nicholas Walczak, Jacob Beal, Jesse Tordoff, and Ron Weiss

Abstract

Laboratory automation now commonly allows high-throughput sample preparation, culturing, and acquisition of microscopy images, but quantitative image analysis is often still a painstaking and subjective process. This is a problem especially significant for work on programmed morphogenesis, where the spatial organization of cells and cell types is of paramount importance. To address the challenges of quantitative analysis for such experiments, we have developed TASBE Image Analytics, a software pipeline for automatically segmenting collections of cells using the fluorescence channels of microscopy images. With TASBE Image Analytics, collections of cells can be grouped into spatially disjoint segments, the movement or development of these segments tracked over time, and rich statistical data output in a standardized format for analysis. Processing is readily configurable, rapid, and produces results that closely match hand annotation by humans for all but the smallest and dimmest segments. TASBE Image Analytics can thus provide the analysis necessary to complete the design-build-test-learn cycle for high-throughput experiments in programmed morphogenesis, as validated by our application of this pipeline to process experiments on shape formation with engineered CHO and HEK293 cells.

Key words Image processing, Cell quantification, Fluorescence microscopy, Programmed morphogenesis, Software tools

1 Introduction

Fluorescence microscopy is one of the most commonly used assay tools of synthetic biology, making use of fluorescent proteins or dyes to deliver rich information about both the state and structure of individual cells and also about the spatial organization of cells, colonies, and tissues. As both protocols and laboratory automation have improved, an increasing number of synthetic biology projects involve high-throughput sample preparation, culturing, and acquisition of microscopy images. With potentially large numbers of wells observed at many different time points, the volume of

fluorescent image data can rapidly become quite large, easily going into the tens or hundreds of gigabytes. This is especially true in the case of work on programmed morphogenesis, where fluorescence images are often used to image the shape of cell cultures and distribution of cell types over time, repeated across a number of different experimental parameters. Yet much of the analysis of image datasets—even quite large ones—is still done qualitatively or by hand. Such analysis is thus typically a time-consuming and painstaking process, as well as subject to a high degree of variability based on observer interpretation. Automation of quantitative analysis using image processing and computer vision techniques can provide great benefits in the use of such data, as well as greatly enhancing the repeatability of these types of experiments.

A number of different image analysis software packages that are specialized to cell biology already exist to aid in this process, such as CellProfiler [1], ImageJ [2, 3], and WIPP [4]. These tools are designed to be broadly applicable to a wide variety of work flows, but require expert crafting by the user to apply them to any particular class of experiments. This makes it difficult for these highly general tools to be applied by researchers who do not simultaneously also have expertise in both software engineering and in developing computer image processing pipelines. Complementarily, a number of specialized packages exist, which are effective but highly tailored for specific purposes, such as SuperSegger [5] (optimized for rod-shaped bacterial cells), NICE [6] (colony counting), or FogBank [7] (overlapping cell segmentation). No such tool, however, had previously been developed for quantifying the shapes of cell populations, as is typically needed for experiments on programmed morphogenesis.

We thusdeveloped this image analysis pipeline to support research in programmed morphogenesis, in the form of a highly configurable pipeline for segmentation and quantification of a broad class of experiments regarding the organization of fluorescent cells in space. We now present the resulting software package, TASBE Image Analytics, distributed under a free and open license at   https://github.com/TASBE/TASBEImageAnalytics.   Our implementation is a processing pipeline developed using the general ImageJ framework, which segments cells and regions of cells in fluorescence microscopy images using a thresholding process, then tracks the evolution of those segments over time. We first describe the architecture and operation of this processing pipeline, then describe its validation by comparison with human annotation, and finally provide an example of its operation in the context of shape formation experiments with engineered CHO and HEK293 cells.

2   Materials

The experiments performed for this work were done in Ubuntu Linux 18.04 using Java 8 and Jython 2.7. The version of ImageJ was 1.52p used from Fiji. The version of TrackMate used was 4.0.1. The data used for experiments presented in this work was captured with a Leica TCS SP5 II Confocal Laser Scanning Microscope. Data generated by a BioTek Cytation5 was also analyzed with this software pipeline. It is our intent that the software could be used with any version of Java and Jython that are compatible with ImageJ although other configurations have not been tested.

3   Methodology

Figure 1 shows the architecture of the TASBE Image Analytics image processing pipeline (named for its relationship with prior automation projects [8, 9]), which is implemented as a set of Jython scripts utilizing ImageJ plugins, proceeding in five stages. First, data and metadata are marshaled to configure the processing. Second, cells and regions of cells are segmented in each microscopy image based on intensity. Third, the binary segment images are filtered to remove artifacts and clustered to identify connected components. Fourth, per-frame statistics are computed for each identified component. Finally, these components can also be tracked through time from one microscopy frame to another.
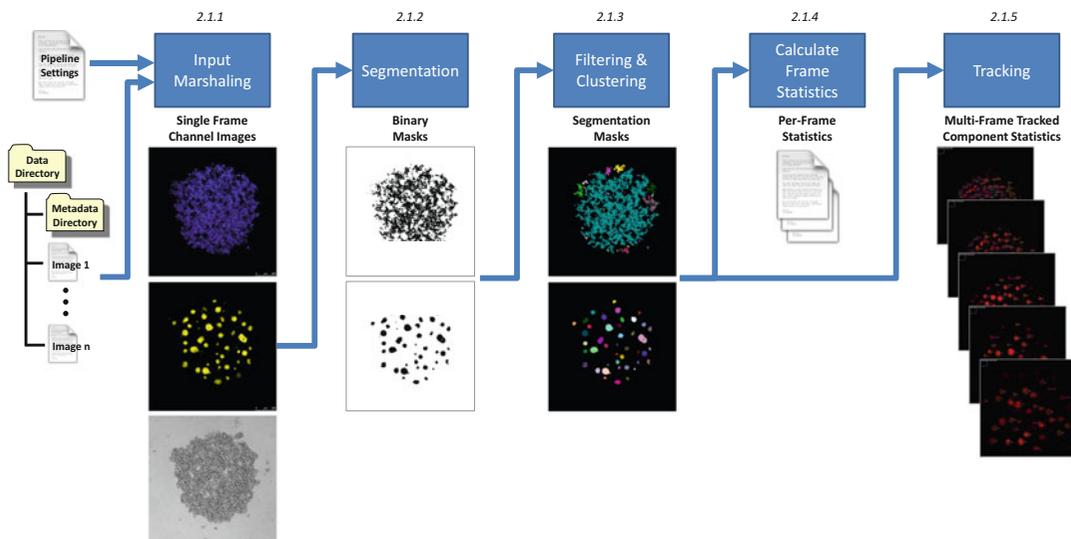
Fig. 1 The TASBE Image Analytics pipeline executes in five stages: (*1*) marshaling microscopy data, metadata, and other configuration settings, (*2*) processing each frame into binary masks, (*3*) filtering and clustering to segment the image, (*4*) calculation of cell cluster statistics, and (*5*) tracking of clusters across timesteps

The steps outlined here represent a common approach to solving this problem. However, this work aims to create a pipeline that is readily available and can work on a wide variety of fluorescence microscopy datasets with a minimum amount of reconfiguration. The nature of the design-build-test-learn cycle, combined with high-throughput sample preparation, means that a large amount of data can be generated in a short period of time, so facilitating quick analysis of the microscopy experiments can allow the cycle times to be shortened. Jython scripts (one of the standard options for scripting ImageJ) were chosen to facilitate this, as they can be run on a directory of microscopy images by just specifying a few parameters in a configuration file.

3.1 Step-by-Step Procedure

3.1.1 Input Marshaling

The first step in the processing pipeline is defining and reading in the data, including metadata that describes information about the input microscopy images. The TASBE Image Analytics pipeline is also highly configurable, with a number of different parameters (intensity threshold parameters, threshold computation algorithm, filtering thresholds, etc.) that can be adjusted by a pipeline settings configuration file. The scripts are also designed to handle directories of microscopy images, as high-throughput microscopes can generally be configured to output files into a structured pattern of directories and filenames for each well in a plate examined by the microscope. Further information about relevant configuration parameters can be found in Subheading 4.2. At present, two instrument-specific classes of metadata are supported: for Leica microscopes, the properties of XML files can be parsed to determine things like number of channels, number of time steps, and number of Z slices, as well as the dimensions of the images in pixels and physical units. Similarly, for BioTek Cytation microscopes, the TIFF tags in the input images can be read to pull out available metadata. For other microscopes, these parameters can be defined manually in the configuration file.

3.1.2 Segmentation

Once the settings and images have been marshaled for processing, the next step is to segment out the foreground of the image from the background. This is done by computing a threshold on the image and only keeping the pixels that meet the threshold. Foreground will be above the threshold for fluorescent images, but for brightfield images an upper threshold is computed as well and only pixels between the two thresholds are kept. Morphological closing [10] is applied to the resulting binary masks, which helps to fill in holes. ImageJ supports numerous different methods for computing the threshold values (default, Otsu, max entropy, and many others, as well as adaptive methods), and these different methods can be specified in the configuration, if desired. Further information about relevant configuration parameters can be found in Subheading 4.3.

3.1.3 Filtering and Clustering

Once the foreground mask is created, the resulting pixels must be grouped together into objects. A common approach to this is connected component analysis, which combines pixels that are touching based on four-connectedness or eight-connectedness [11]. An advanced version of this approach is performed with ImageJ's ParticleAnalyzer tool, which also allows the resulting objects to be filtered based on several parameters, such as circularity. Further information about relevant configuration parameters can be found in Subheading 4.4.

3.1.4 Calculation of Frame Statistics

A set of statistics is then output for each object detected in each frame, including centroid, height, width, perimeter, area (in pixels^2), area (in microns^2), and many other standard image component statistics. Images of the segmentation masks at each stage are also produced in order to help debug processing.

3.1.5 Tracking

Once cell clusters have been identified at each timestep, their identities need to be associated across time such that the progression of each cell is tracked. There are several tracking plugins available for ImageJ, of which we have selected TrackMate [12], a recent addition that offers a powerful and flexible interface. We combine TrackMate with the previously described threshold-based detection mechanism to implement multi-frame tracking. The result is another set of statistics, summarizing all tracking information for all of the components in each frame.

Once configured, the execution of the complete processing pipeline is quite fast, even on substantial high-throughput datasets. Because TASBE Image Analytics is built as an application of mature image processing tools, it is able to operate quickly and efficiently. We benchmarked performance by processing 1024×1024 images with three channels on an Intel i7 equipped laptop, finding that each image took an average of only 2.2 s to process.

3.2 Validation

We validated the performance of TASBE Image Analytics against hand-labeled ground truth by comparison of detections for a collection of 60 microscopy images. Hand-labeling was done with an interactive labeling script created in Python using the GrabCut [13] implementation in OpenCV [14], allowing a human to draw a rectangle around a region of interest and then mark some background and foreground pixels to generate a segmentation mask.

Human and machine labeling are compared with a standard widely used metric [15]: bounding boxes B_h and B_m, determined by human and machine, respectively, are compared using intersection over union (IOU):

$$\text{IOU}(B_h, B_m) = \frac{\text{Area}(B_h \cap B_m)}{\text{Area}(B_h \cup B_m)}, \tag{1}$$

judging two components sufficiently equivalent when IOU is greater than or equal to 50%. This allows performance to be judged in terms of true positives (TP, equivalent components), false negatives (FN, human segment with no machine equivalent), and false positives (FP, machine segment with no human equivalent), from which we compute standard [16] performance metrics precision, recall, and F_1 score:

$$\text{Precision} = TP/(TP + FP), \tag{2}$$

$$\text{Recall} = TP/(TP + FN), \tag{3}$$

$$F_1 = 2TP/(2TP + FN + FP). \tag{4}$$

Our evaluation used 60 images (1110 labeled components) from a CHO and HEK293 coculture experiment, ignoring any component with an area less than 350 microns^2. Overall performance was satisfactory, with a total recall of 82.3%, total precision of 97.1%, and a total F_1 of 89.1%. More importantly, nearly all errors involved small components (statistics by area range are presented in Fig. 2a), which tend to have weaker fluorescent returns and hence can sometimes dip below the automatically computed thresholds, as well as being more sensitive to small differences in edge location. In many cases, these issues can also be mitigated by choosing a different threshold computation algorithm or specifying a default threshold to fall back on if the automated threshold is problematic. TASBE Image Analytics may thus be expected to provide human-level performance in segmenting microscopy images.

3.3 Example Results   To illustrate the use of this method, we show an example of how the TASBE Image Analytics pipeline has been applied experimentally to quantification of microscopy data from shape-formation experiments. These experiments considered mixtures of CHO and HEK293 cells, genetically modified to express different fluorescent proteins and using differential levels of cadherin expression to sort into various spatial patterns (full details of this work may be found in [17]). Figure 2b–d shows samples of results produced using TASBE Image Analytics from an experiment in which mixtures of HEK293 and CHO were imaged every 20 min over the course of 66 hours, with one 68-min gap around hour 13.

The rich collection of statistics generated from the TASBE Image Analytics image processing pipeline can be plotted and used in various ways to draw conclusions about the experiment. For example, in the case of these CHO/HEK293 adhesion experiments, it was predicted that low concentrations of HEK293 cells would result in the formation of a multiple cluster pattern. By plotting the areas of components over time at different concentrations, we were able to visually validate this hypothesis (Fig. 2b), as well as quantitatively validate the hypotheses through analysis of the statistics produced from those images.

(a)

(b)

(c)

(d)
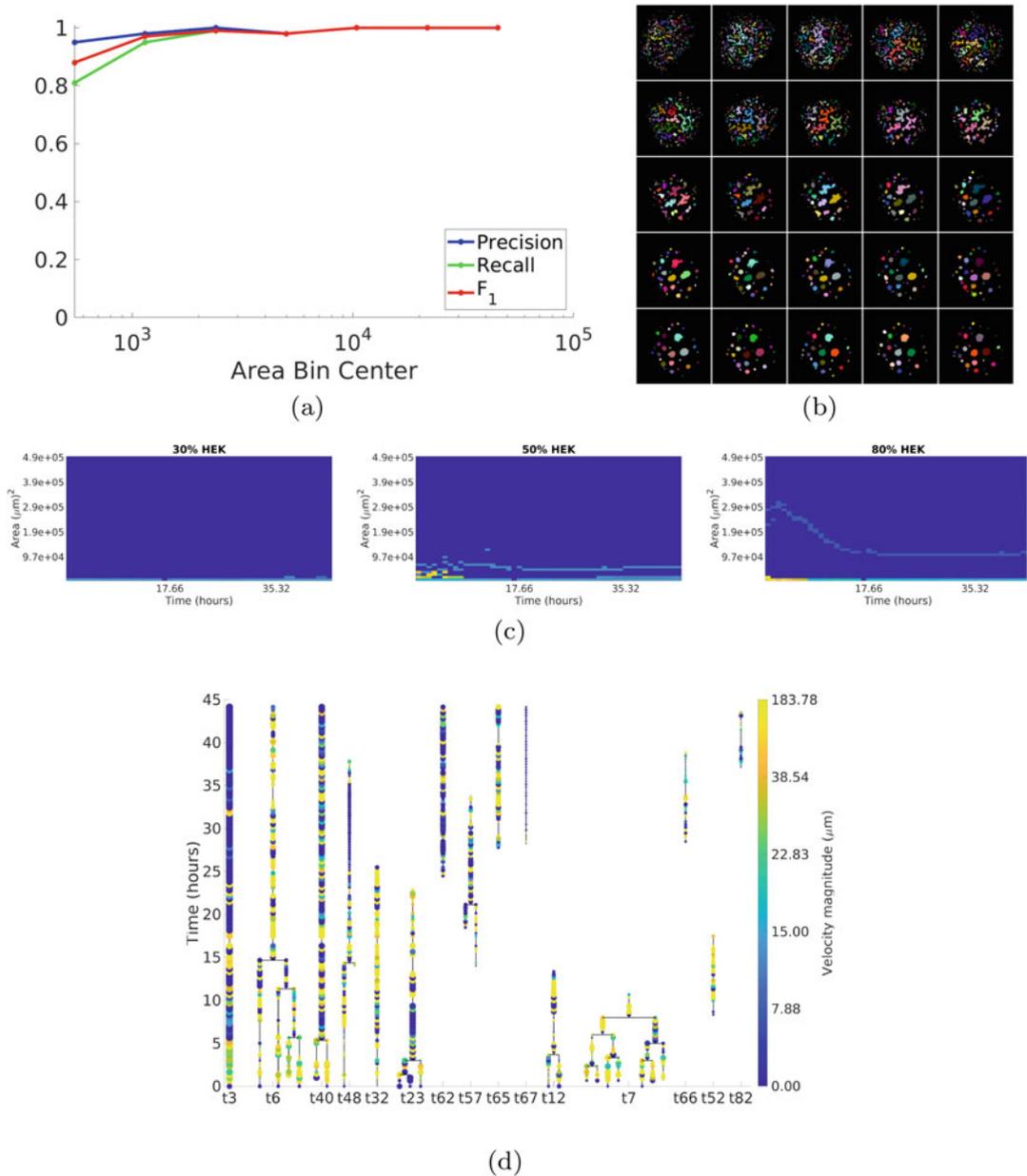
Fig. 2 Validation and experimental results: (**a**) Processing images with TASBE Image Analytics provides results closely equivalent to hand processing by humans, particularly for larger components. (**b**) Segmentation showing formation of a "polka dot" pattern in a mixture of HEK and CHO cells over time (time progresses left to right then top to bottom). (**c**) Heat maps of component size vs. time showing a transition from small fast-moving components at 30% HEK to a single large slow-moving component at 80% HEK (warmer colors are faster, dark blue means no components have that area at that time). (**d**) Tracking "phylogeny tree" showing how smaller components combine to form larger components over time

Figure 2c shows another example of a result computed from statistics over time. Here, the color of each cell in the heat map corresponds to the mean velocity of all components within an area range and time period. From this plot, we can see that at 30% HEK293, there are only small components, however around 50% a phase transition begins, where some larger components form, and by 80% there is a large component that forms (condensing to a smaller area) and then grows over time. In addition, we can see that smaller components move faster than large components in this data. Figure 2d shows one more example of the use of tracking, in this case a "phylogeny" tree graph that shows how smaller components combine to form larger components over time, as well as the area (dot size) and velocity (color) of these components.

These examples are by no means exhaustive: they merely illustrate a few of the many ways in which TASBE Image Analytics can be applied to data from real programmed morphogenesis experiments in order to provide insight and quantification.

4   Notes

4.1   Software Setup

The code necessary to run our pipeline can be found in the TASBE organization on GitHub: https://github.com/tasbe. There are three related repositories: TASBEImageAnalytics, TASBEImageAnalytics-Tutorial, and TASBEImageAnalytics-Data. The first repository houses the source code including Jython scripts for running the processing pipeline, Java code to create a thresholding-based detector for Track-Mate, and C++ programs for creating point clouds from z-stacks generated by a confocal microscope (an aspect not covered in the main methods description above). The tutorial repository contains some shell scripts that illustrate how to execute the image analysis pipeline, and which can be used as a template for configuration of the pipeline for new experiments. The data repository, in turn, contains example image data used by the tutorial repository scripts.

In order to use the pipeline, one must download the source code and install ImageJ. For all of our processing, we used the ImageJ distribution FIJI (https://fiji.sc/). The scripts in the tutorials repository give a way to use the processing pipeline, and the data in the data repository show a common layout for the microscopy experiments we have worked with.

4.2   Parameter Configuration

Tables 1, 2, and 3 define all of the parameters recognized in the configuration file. These parameters are split into three groups: control of filename parsing, configuration of dataset properties, and configuration for processing. The filename parsing is important so that all of the files are properly marshaled. Data is grouped together by well on the plate, and across the possible channels, timesteps, and Z slices. Frequently, this information is encoded in

Table 1
Table of configuration parameters recognized in the configuration .ini file

Filename parsing options		
Parameter	**Name in File**	**Description**
Directory of microscopy images	inputDir	Directory that contains all of the microscopy images to process.
Location of results/output	outputDir	Directory where all outputs will be stored.
Well Index	wellIdx	Index of the well name in the filename when split on "_". Can be a comma separated list of values. Required.
Channel Index	cIdx	Index of the channel specifier in the filename when split on "_". Can be detected if the token has "ch" in it.
Z Index	zIdx	Index of the Z slice specifier in the filename when split on "_". Can be detected if the token has "z" in it.
Time Index	tIdx	Index of the timestep specifier in the filename when split on "_". Can be detected if the token has "t" in it.
Specify wells to process	wellNames	If specified, only wells in the comma separated list will be processed.
Specify wells to ignore	excludeWellNames	If specified, any wells in the comma separated list will be skipped.

This section contains options that determine how the filenames are parsed

the filename, and the script is able to extract this information when the tokens are separated by underscores ("_"). Well names generally need to be specified, but channel, timestep, and Z slice can be found automatically if their tokens contain "ch," "t," and "z" selectively, as are often used in microscopy filenames. Most of the dataset parameters can be found in microscope property files. For Leica confocal microscopes, these are contained in a metadata directory as an xml file. All of the data parsed from the properties files can be specified manually, but utilizing the xml files cuts down on the amount of configuration that is necessary.

4.3 Threshold Parameters

The last set of parameters, the processing parameters, are the ones that have the most effect on the outputs. If the computed threshold is too low, it can lead to a lot of background noise being considered and generally yields a poor result. To counter this, the maximum and minimum thresholds computed by ImageJ are compared: if the difference is too high, then the computed threshold is replaced by the default threshold. This is controlled by the maxThreshRange and defaultThreshold parameters. The FIJI distribution of ImageJ contains over a dozen different methods for computing an intensity threshold, and different algorithms can yield different results. The

Table 2
Table of configuration parameters recognized in the configuration .ini file

Dataset property options		
Parameter	**Name in File**	**Description**
Number of Channels	numChannels	Specifies number of channels in input. Read from microscope properties or set manually.
Number of Z slices	numZ	Specifies number of z slices in input. Read from microscope properties or set manually.
Number of timesteps	numT	Specifies number of timesteps in input. Read from microscope properties or set manually.
The first timestep	minT	Specifies the first timestep to start on. Defaults to 0.
No Z in Filename	noZInFile	Flag that indicates filenames do not contain Z slice specifiers.
No T in Filename	noTInFile	Flag that indicates filenames do not contain timestep specifiers.
Labels for channels	chanLabel	Comma separated list to label channels. Default is [Skip, Yellow, Blue, Gray].
Channels to skip	chansToSkip	List of channel labels for channels that should be ignored/skip. A channel labeled Skip will also be skipped.
Physical size of pixels	pixelHeight	Defines physical height of each pixel. Read from microscope properties. If not specified then areas will be in the value of pixels.
Physical depth of pixel	pixelDepth	Defines physical depth of each pixel. Read from microscope properties. If not specified then areas will be in the value of pixels.
Physical width of pixel	pixelWidth	Defines physical width of each pixel. Read from microscope properties. If not specified then areas will be in the value of pixels.
Debug mode flag	debugOutput	If specified additional output will be created to help with debugging. Optional.
Lower right exclusion X	lowerRightExclusionX	X coord for box to exclude in the lower right, where scale bars commonly appear. Optional.
Lower right exclusion Y	lowerRightExclusionY	Y coord for box to exclude in the lower right, where scale bars commonly appear. Optional.
Upper left exclusion X	upperLeftExclusionX	X coord for box to exclude in the upper left, where timestamps commonly appear. Optional.
Upper left exclusion Y	upperLeftExclusionY	Y coord for box to exclude in the upper left, where timestamps commonly appear. Optional.

This section contains options that specify properties of the dataset, most of which can be read from microscope property files

Table 3
Table of configuration parameters recognized in the configuration .ini file

Processing options		
Parameter	**Name in File**	**Description**
Max threshold range	maxThreshRange	Used to define a maximum range between the computed upper and lower thresholds. If exceeded the image is ignored or a default threshold is used.
Default threshold	defaultThreshold	Controls what happens if maximum threshold range is exceeded. If not set, image is ignored otherwise the set value is used as the threshold.
Thresholding method	thresholdMethod	Defines which method will be used to compute the image threshold.
Threshold from whole range	thresholdFromWholeRange	Only works for tracking, if true image threshold will be computed from all images instead of each frame.
Area threshold—% of max	areaMaxPercentThreshold	A threshold on area to remove unwanted cell clusters, defined as a percentage of the maximum area found in the current frame.
Area threshold—absolute	areaAbsoluteThreshold	A threshold on area to remove unwanted cell clusters, defined as an absolute area value.
Create segmentation masks	createSegMask	If specified, the outputs will include segmentation mask images where pixel values denote blob membership. This option does increase runtime.
LUT Path	lutPath	Specify a file to use as the LUT for segmentation masks. Controls the colors used for each detected cell cluster.

This section contains options that affect the output of processing

method that is used can be specified by the thresholdMethod parameter, although the default value works for many cases. FIJI has a good way to see the results of all available thresholding algorithms on a single image by using Image>Adjust>Auto Threshold. . . If the Try All method is used, FIJI will display the results for each image in a single collage. Finally, the default is to compute a threshold for each image independently of the other images. In some cases, it can be better to compute a single threshold to use on all images from the image data contained in all of the images. This can be enabled using the thresholdFromWholeRange option, though this option currently only works for the cellStatsTracking script.

4.4 Cluster Parameters

In some cases, some of the detected cell clusters are too small to include in data analysis. There are two parameters provided that can help to remove some of the smaller and more transient detections. The first one, areaAbsoluteThreshold, can be used to remove any

cell cluster with an area smaller than a defined threshold. The second parameter, areaMaxPercentThreshold, attempts to scale the threshold parameter by thresholding on a percentage of the largest cluster in the current frame.

4.5 Debugging Parameters

The createSegMask parameter can be useful for debugging results, and can also be used to apply the cell cluster segmentation in other contexts. When true, segmentation mask images will be created where each pixel in the output image will identify which cluster the pixel belongs to. Each cluster will be uniquely colored, and the color used is defined by a look-up table (LUT), which is defined by the lutPath parameter (FIJI comes with several different LUT options).

By adjusting these parameters, a large number of different situations can be covered. We have demonstrated that TASBE Image Analytics provides a high-throughput processing pipeline to segment cells and regions of cells in microscopy images and to track them over time. This processing pipeline has been validated against hand-labeled data and its utility has been demonstrated in quantifying experiments on shape formation with engineered CHO and HEK293 cells. We have made this system available under a permissive open-source license in the hopes that it will prove useful for a broad range of experiments involving fluorescent cells. Future development is anticipated to be incremental maintenance, refinement, and generalization as driven by the evolving needs of additional users and applications.

Acknowledgments

References

1. Lamprecht MR, Sabatini DM, Carpenter AE (2007) Cellprofiler: free, versatile software for automated biological image analysis. BioTechniques 42(1):71–75

2. Rasband W (2012) Imagej: image processing and analysis in java. Astrophys Sour Code Lib 1:06013

3. Schindelin J, Rueden CT, Hiner MC, Eliceiri KW (2015) The imagej ecosystem: an open platform for biomedical image analysis. Mol Reprod Dev 82(7-8):518–529

4. Bajcsy P, Chalfoun J, Simon MH (2018) Web microanalysis of big image data. Springer International Publishing, New York

5. Stylianidou S, Brennan C, Nissen SB, Kuwada NJ, Wiggins PA (2016) Supersegger: robust image segmentation, analysis and lineage

tracking of bacterial cells. Mol Microbiol 102 (4):690–700

6. Clarke ML, Burton RL, Hill AN, Litorja M, Nahm MH, Hwang J (2010) Low-cost, high-throughput, automated counting of bacterial colonies. Cytom Part A 77(8):790–797

7. Chalfoun J, Majurski M, Dima A, Stuelten C, Peskin A, Brady M (2014) Fogbank: a single cell segmentation across multiple cell lines and image modalities. BMC Bioinformatics 15 (1):431

8. Beal J, Weiss R, Densmore D, Adler A, Appleton E, Babb J, Bhatia S, Davidsohn N, Haddock T, Loyall J, Schantz R, Vasilev V, Yaman F (2012) An end-to-end workflow for engineering of biological networks from high-level specifications. ACS Synth Biol 1 (8):317–331

9. Beal J, Overney C, Adler A, Yaman F, Tiberio L, Samineni M (2019) Tasbe flow analytics: a package for calibrated flow cytometry analysis. ACS Synth Biol 8(7):1524–1529

10. Efford N (2000) Digital image processing: a practical introduction using java (with CD-ROM), 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA

11. Forsyth DA, Ponce J (2002) Computer vision: a modern approach. Professional Technical Reference, Prentice Hall

12. Tinevez JY, Perry N, Schindelin J, Hoopes GM, Reynolds GD, Laplantine E, Bednarek SY, Shorte SL, Eliceiri KW (2017) Trackmate: an open and extensible platform for single-particle tracking. Methods 115:80–90

13. Rother C, Kolmogorov V, Blake A (2004) Grabcut: interactive foreground extraction using iterated graph cuts. ACM Trans Graph 23:309–314

14. Bradski G (2000) The openCV library. Dr Dobb's J Softw Tools 120:122–125

15. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. Int J Comput Vis 88(2):303–338

16. Tan PN, Steinbach M, Kumar V (2005) Introduction to data mining, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA

17. Tordoff J, Krajnc M, Walczak N, Lima M, Beal J, Shvartsman S, Weiss R (2020) Incomplete cell sorting creates engineerable structures with long term stability. Cell Reports Physical Science