Mother machine image analysis with MM3

John T. Sauls [†] * Jeremy W. Schroeder [‡] * Steven D. Brown [†] Guillaume Le Treut [†]
Fangwei Si [†] Dongyang Li [†] Jue D. Wang [‡] Suckjoon Jun [† §]

The mother machine is a microfluidic device for high-throughput time-lapse imaging of microbes. Here, we present MM3, a complete and modular image analysis pipeline. MM3 turns raw mother machine images, both phase contrast and fluorescence, into a data structure containing cells with their measured features. MM3 employs machine learning and non-learning algorithms, and is implemented in Python. MM3 is easy to run as a command line tool with the occasional graphical user interface on a PC or Mac. A typical mother machine experiment can be analyzed within one day. It has been extensively tested, is well documented and publicly available via Github.

Introduction

Image analysis is a major bottleneck for live-cell, high-throughput, time-lapse imaging. The mother machine is a popular microfluidic platform for such imaging[1]. This brief introduces an image analysis pipeline for mother machine experiments named MM3. MM3 is implemented in Python and can be accessed at https://github.com/junlabucsd/mm3 along with guides and a Docker container.

There are a number of mother machine-specific image analysis packages available[2–9] in addition to general cell image analysis packages which may be adapted to a mother machine workflow[10–13]. MM3 aims to be a complete and flexible solution to this problem, taking raw micrographs and producing readily graphable cell data. It includes support for phase contrast and fluorescence images, and has been tested with different species (bacteria and yeast), mother machine designs, and optical configurations. We refer potential users to our recent publications for examples of the use of MM3[14,15].

The pipeline architecture is modular, which allows flexible use of mid-stream outputs and straightforward troubleshooting. Time-lapse image analysis of this nature is normally split into two tasks: segmentation and tracking of cells[12]. MM3 provides standard and supervised machine learning options for both. Graphical User Interfaces (GUIs) are included for creating training data for machine learning methods. The methods and documentation herein describe the pipeline generally. Users are encouraged to find the most up-to-date documentation in the GitHub code repository.

How MM3 works

Image analysis via MM3 can be thought of in four discrete modules Figure **1**. Raw images are cropped and compiled into stacks corresponding to individual growth channels. Channel stacks are then presented to the user to be selected for analysis or ignored. Cells are then segmented from each other. Finally, segments are tracked through time to create cell lineages. Individual cells in the lineages are represented as objects which can be plotted directly or converted to another data format.

Each module comprises multiple methods from which the user may choose, usually one non-learning and one supervised learning. Generally speaking, the non-learning methods are faster but optimized for regular mother machine designs and phase contrast imaging of bacteria. The supervised learning methods can be applied to a wider range of experimental set-ups and have low error rates, but require annotated training data. A common strategy is to use the non-learning methods to seed training data for the learning methods.

1 Channel compilation and designation

The first section of the MM3 pipeline consumes raw micrographs and returns image stacks corresponding to one growth channel through time [¶]. Further pipeline operations are then applied to these stacks.

A standard mother machine experiment consists of thousands of images across multiple fields of view (FOVs) and many time points. Images are first collated based on the available metadata. MM3 expects TIFF files, and looks for metadata in the TIFF header and from the file name.

All images from a particular FOV are analyzed for the location of channels using the phase contrast plane. Channel detection may be performed by using either a wavelet transform or a convolution neural network. In the former method of channel detection, a mask is made which is applied across all time points. In the latter channel detection method, a separate mask is generated for each timepoint. This enables accurate alignment of channels when large amounts of stage drift is an issue. Channels are cropped through time using the masks and saved as unique image stacks that include all timepoints for a given channel and imaging plane. MM3 saves channel stacks in TIFF format.

MM3 attempts to compile all channels. However, not all channels contain cells and some channels may have undesirable artifacts from the device preparation. It is therefore desirable to only process certain channels for analysis. MM3 implements several methods for the user to choose from to identify which channels should be analyzed. In our expe-

* These authors contributed equally to this work.
[†] Department of Physics, University of California San Diego, La Jolla, CA
[§] Section of Molecular Biology, Division of Biology, University of California San Diego, La Jolla, CA
[‡] Department of Bacteriology, University of Wisconsin-Madison, Madison, WI

[¶] We refer to growth channels as 'channels,' while additional images at the same time point (i.e., an image channel) as 'planes.'

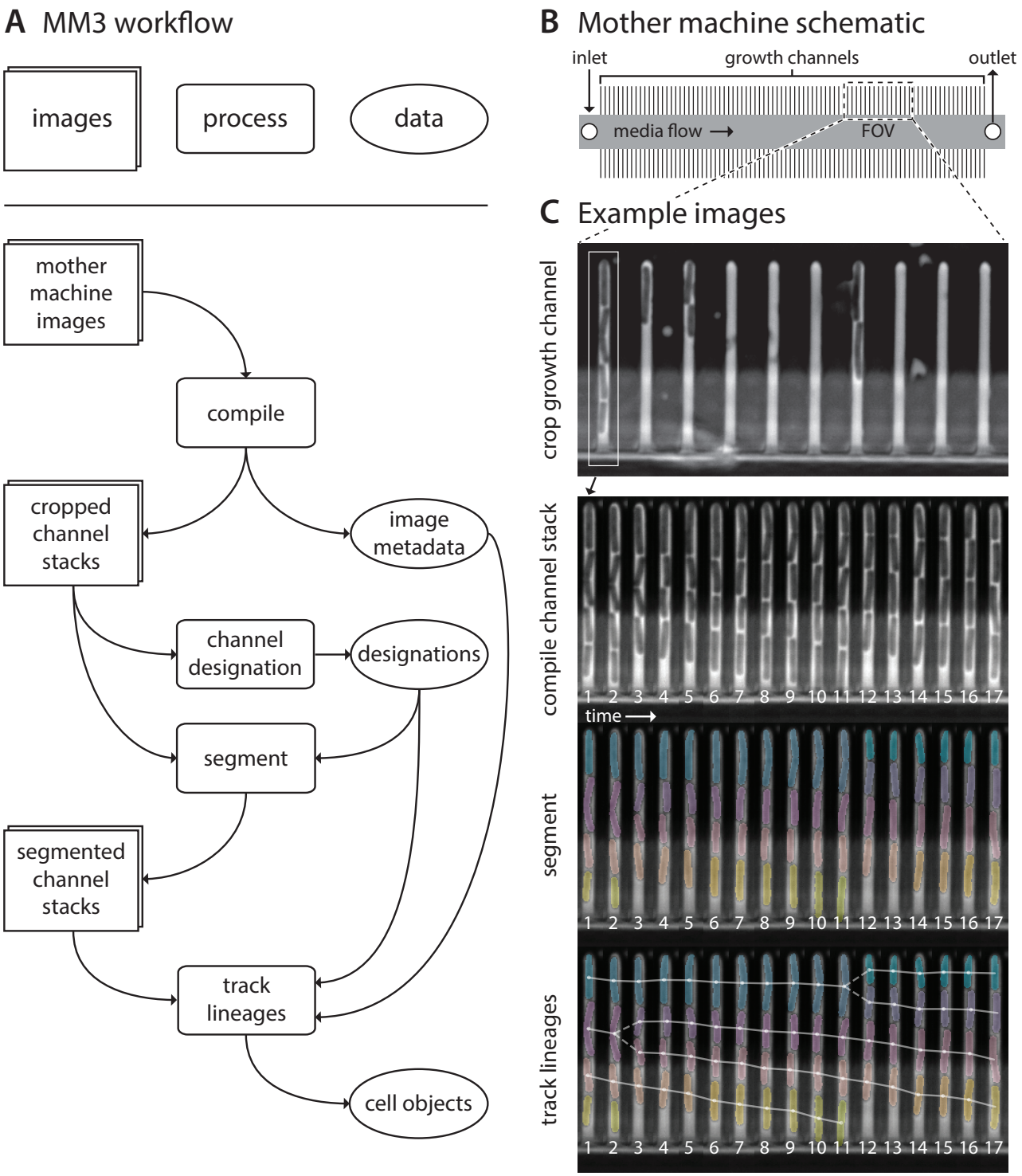

Figure 1: **MM3 workflow and example images.** (A) The MM3 image analysis pipeline takes raw mother machine images and produces cell objects. Processes (rounded rectangles) are modular; multiple methods are provided for each. (B) Mother machine device schematic. Growth channels flank a central flow cell which supplies fresh media and whisks away daughter cells. In a typical experiment numerous fields of view (FOV) are imaged for several hours. (C) Example images from the processing of one growth channel in a single FOV. The growth channel is first identified, cropped, and compiled in time. All cells are segmented (colored regions). Lineages are tracked by linking segments in time to determine growth and division (solid and dashed lines, respectively), creating cell objects.

rience it is prudent to use the included GUI to manually designate channels to retain for data analysis.

2 Cell segmentation

Cell segmentation is the first of the two major tasks in the image analysis pipeline. Segmentation receives channel stacks and produces 8-bit segmented image stacks. Typically, segmentation is done using the phase contrast time-collated stack.

MM3 has two methods for segmentation: a "standard" method and a supervised learning method. The standard method uses traditional image analysis techniques, specif-

ically background subtraction, Otsu thresholding, morphological operations, and watershedding/diffusion. The supervised learning method uses a convolution net implement using the U-net architecture[16]. A GUI and Jupyter notebook is provided to both annotate training data and create the convolution net model.

Illumination conditions can vary across laboratories, microbial species, and with device design. To improve performance of the U-net on specific conditions, we recommend a strategy which uses standard morphological techniques to generate segmentation data for training. A subset of this segmentation data is subsequently manually curated using the provided GUI tool for use as U-net training data. U-net training refinement on these datasets can be done relatively fast, so subsets of training data can quickly be tested for performance.

3 Cell tracking

Tracking segmented cells is the second major task in the pipeline. Tracking involves linking cell segments in time in order to define a lineage of cell objects. Two methods are provided for tracking. One is a simple decision tree based on *a priori* knowledge of binary fission and the mother machine. For example, cells normally grow by a small amount between time intervals, divide into two similarly sized daughter cells, and cannot pass each other in the channel.

The second tracking algorithm applies several learned classifiers to cell segmentation data to assign the probability that each cell, at a given time point i, appeared in i, was born in i, disappeared after i, dies after i, is the parent of each cell in i+1, and is the same cell as each cell in i+1. The results of running these models on windows of time in the segmented image stacks are used to determine the most likely track paths through a graph in which cells are nodes and probabilities of each event described above are edges. A GUI is provided to create annotated track data for training. As with segmentation, the non-learning method can be used as a foundation for creating high-quality training data for the learning method.

4 Data output and analysis

Tracking produces a dictionary of cell objects which contains relevant information derived from the cell segments. This includes, but is not limited to, birth and division size, growth rate, and generation time. Each object is identified by a key which represents the FOV and channel of the cell, the time point of its birth, and its position in the channel.

Since each cell object has the requisite information to find its corresponding position in the channel stacks, the objects can be appended via additional analysis. For example, the corresponding location of a cell in a fluorescent image stack can be retrieved, focus detection performed, and that information can be added to the cell object. This minimizes the burden of rerunning previous sections of the pipeline for new sub-analyses.

MM3 indeed includes methods for fluorescence analysis, from simple quantification of the fluorescence signal to foci tracking[14].

Plotting can be done from this cell object dictionary directly, or it can first be converted to a .csv, a pandas DataFrame, or a Matlab structure.

Availability, implementation, performance, and guides

MM3 is primarily run as a command line tool, and comprises several scripts. GUIs are provided for curating cell segmentation training data and cell tracking training data, in addition to curating which channels to include in final analyses. Parameters are passed to the individual scripts via a text file in YAML format and as command line options. It is readily forked or downloaded from the GitHub repository (https://github.com/junlabucsd/mm3).

MM3 uses scikit-image[17] for image analysis methods, TensorFlow[18] for learning methods including U-net segmentation, and NetworkX[19] for tracking graphs. MM3 is indebted to the "SciPy ecosystem" (SciPy, NumPy, Jupyter/IPython, Matplotlib, and pandas[20–24]).

MM3 will run or a standard PC, Mac, or Linux machine, and a Docker container is provided to ameliorate installation headaches. A mother machine experiment consisting of 30Gb of raw image data (18 hours, 2 imaging planes, 50 FOVs, 25 growth channels per FOV) can be analyzed in less than one day in a Docker container using 24Gb of RAM with a 3.5 GHz Intel Core i7 and no GPU.

Significant documentation exists on the MM3 repository, covering both installation and usage. MM3 is intended to be accessible for those with minor programming experience, such as an undergraduate physics or engineering student. For more advanced users, the modular framework is well-suited to develop alternative approaches for certain tasks.

Download MM3: https://github.com/junlabucsd/mm3

User Forum: https://piazza.com/ucsd/fall2019/mm3

References

[1] "Robust growth of Escherichia coli.," *Current biology*, 20, 1099–103, 2010.

[2] M. Arnoldini, I. A. Vizcarra, R. Peña-Miller, N. Stocker, M. Diard, V. Vogel, R. E. Beardmore, W.-D. Hardt, & M. Ackermann, "Bistable Expression of Virulence Genes in Salmonella Leads to the Formation of an Antibiotic-Tolerant Subpopulation.," *PLoS Biology*, 12, p. e1001928, 2014.

[3] F. Jug, T. Pietzsch, D. Kainm, J. Funke, M. Kaiser, E. V. Nimwegen, C. Rother, & G. Myers, "Optimal Joint Segmentation and Tracking of Escherichia Coli in the Mother Machine," 25–36, 2014.

[4] *Time-lapse microscopy and image analysis of Escherichia coli cells in mother machines*, vol. 43. Elsevier Ltd., 1 ed., 2016.

[5] C. C. Sachs, A. Grunberger, S. Helfrich, C. Probst, W. Wiechert, D. Kohlheyer, & K. Nöh, "Image-based single cell profiling: High-throughput processing of mother machine experiments," *PLoS ONE*, 11, 1–15, 2016.

[6] M. Kaiser, F. Jug, T. Julou, S. Deshpande, T. Pfohl, O. K. Silander, G. Myers, & E. van Nimwegen, "Monitoring single-cell gene regulation under dynamically controllable conditions with integrated microfluidics and software," *Nature Communications*, 9, p. 212, 2018.

[7] J.-B. Lugagne, H. Lin, & M. J. Dunlop, "DeLTA: Automated cell segmentation, tracking, and lineage reconstruction using deep learning," *bioRxiv*, 1–17, 2019.

[8] J. Ollion, M. Elez, & L. Robert, "High-throughput detection and tracking of cells and intracellular spots in mother machine experiments.," *Nature protocols*, 2019.

[9] A. Smith, J. Metz, & S. Pagliara, "MMHelper: An automated framework for the analysis of microscopy images acquired with the mother machine," *Scientific Reports*, 9, p. 10123, 2019.

[10] S. Stylianidou, C. Brennan, S. B. Nissen, N. J. Kuwada, & P. A. Wiggins, "SuperSegger : robust image segmentation, analysis and lineage tracking of bacterial cells," *Molecular Microbiology*, 102, 690–700, 2016.

[11] S. K. Sadanandan, O. Baltekin, K. E. G. Magnusson, A. Boucharin, P. Ranefall, J. Jalden, J. Elf, & C. Wahlby, "Segmentation and Track-Analysis in Time-Lapse Imaging of Bacteria," *IEEE Journal of Selected Topics in Signal Processing*, 10, 174–184, 2016.

[12] V. Ulman, M. Maška, K. E. G. Magnusson, O. Ronneberger, C. Haubold, N. Harder, P. Matula, P. Matula, D. Svoboda, M. Radojevic, I. Smal, K. Rohr, J. Jaldén, H. M. Blau, O. Dzyubachyk, B. Lelieveldt, P. Xiao, Y. Li, S. Y. Cho, A. C. Dufour, J. C. Olivo-Marin, C. C. Reyes-Aldasoro, J. A. Solis-Lemus, R. Bensch, T. Brox, J. Stegmaier, R. Mikut, S. Wolf, F. A. Hamprecht, T. Esteves, P. Quelhas, Ö. Demirel, L. Malmström, F. Jug, P. Tomancak, E. Meijering, A. Muñoz-Barrutia, M. Kozubek, & C. Ortiz-De-Solorzano, "An objective comparison of cell-tracking algorithms," *Nature Methods*, 14, 1141–1152, 2017.

[13] D. Bannon, E. Moen, E. Borba, A. Ho, I. Camplisson, B. Chang, E. Osterman, W. Graf, & D. V. Valen, "DeepCell 2.0: Automated cloud deployment of deep learning models for large-scale cellular image analysis," *bioRxiv*, p. 505032, 2018.

[14] F. Si, G. Le Treut, J. T. Sauls, S. Vadia, P. A. Levin, & S. Jun, "Mechanistic Origin of Cell-Size Control and Homeostasis in Bacteria," *Current Biology*, 29, 1760–1770, 2019.

[15] J. T. Sauls, S. E. Cox, V. Castillo, Z. Ghulam-Jelani, & S. Jun, "Gram-positive and Gram-negative Bacteria Share Common Principles to Coordinate Growth and the Cell Cycle at the Single-cell Level," *bioRxiv*, 2019.

[16] O. Ronneberger, P. Fischer, & T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, vol. 9351, 234–241, 2015.

[17] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, & scikit-image contributors, "scikit-image: image processing in python," *PeerJ*, 2, p. e453, 2014.

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, & Others, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283, 2016.

[19] A. Hagberg, P. Swart, & D. S Chult, "Exploring network structure, dynamics, and function using NetworkX," tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[20] E. Jones, T. Oliphant, P. Peterson, & Others, "SciPy: Open source scientific tools for python," 2001.

[21] S. Van Der Walt, S. C. Colbert, & G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Comput. Sci. Eng.*, 13, p. 22, 2011.

[22] F. Perez & B. E. Granger, "IPython: A system for interactive scientific computing," 2007.

[23] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, 9, 90–95, 2007.

[24] W. McKinney & Others, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, 51–56, 2010.

Acknowledgements This work was supported by the Paul G. Allen Family Foundation, Pew Charitable Trust, NSF CAREER grant MCB-1253843, NIH grant R01 GM118565-01 (to S.J.), and NIH grant T32GM8806 (to J.T.S).

Author Information The authors declare no competing financial interests.