Tools and methods for high-throughput single-cell imaging with the mother machine

Ryan Thiermann[1]*, Michael Sandler[1]*, Gursharan Ahir[1]*, John T. Sauls[1]*, Jeremy W. Schroeder[2]*, Steven D. Brown[1], Guillaume Le Treut[3], Fangwei Si[4], Dongyang Li[5], Jue D. Wang[6], Suckjoon Jun[1]**

[1] Department of Physics, University of California San Diego, La Jolla CA

[2] Department of Biological Chemistry, University of Michigan Medical School, Ann Arbor, MI

[3] Chan Zuckerberg Biohub, San Francisco, CA

[4] Department of Physics, Carnegie Mellon University, Pittsburgh, PA

[5] Division of Biology and Biological Engineering, California Institute of Technology, Pasadena, CA

[6] Department of Bacteriology, University of Wisconsin-Madison, Madison, WI

*These authors contributed equally to this work

**Corresponding author: suckjoon.jun@gmail.com

Abstract

Despite much progress, image processing remains a significant bottleneck for high-throughput analysis of microscopy data. One popular platform for single-cell time-lapse imaging is the mother machine, which enables long-term tracking of microbial cells under precisely controlled growth conditions. While several mother machine image analysis pipelines have been developed in the past several years, adoption by a non-expert audience remains a challenge. To fill this gap, we implemented our own software, MM3, as a plugin for the multidimensional image viewer napari. napari-MM3 is a complete and modular image analysis pipeline for mother machine data, which takes advantage of the high-level interactivity of napari. Here, we give an overview of napari-MM3 and test it against several well-designed and widely-used image analysis pipelines, including BACMMAN and DeLTA. In addition, the rapid adoption and widespread popularity of deep-learning methods by the scientific community raises an important question: to what extent can users trust the results generated by such "black box" methods? We explicitly demonstrate "What You Put Is What You Get" (WYPIWYG); i.e., the image analysis results can reflect the user bias encoded in the training dataset. Finally, while the primary purpose of this work is to introduce the image analysis software that we have developed over a decade in our lab, we also provide useful information for those who want to implement mother-machine-based high-throughput imaging and image analysis methods in their research. This includes our guiding principles and best practices to ensure transparency and reproducible results.

Introduction

The mother machine [1] is a popular microfluidic platform for long-term, high-throughput imaging of single cells. It has been widely adopted as a standard for long-term imaging of bacteria such as *Escherichia coli* and *Bacillus subtilis* [2], as well as eukaryotes *Saccharomyces cerevisiae* [3] and *Schizosaccharomyces pombe* [4]. In the mother machine, thousands of single cells are trapped in one-ended growth channels that open into a central trench (Figure 1.1). The cells at the end of the growth channels ("mother cells") grow and divide over hundreds of generations, while their progeny are successively flushed out of the device (Figure 1.2-1.3). Data gathered from the mother machine has brought critical insight into diverse domains such as aging [1], single-cell physiology [5], starvation adaptation [6], antibiotic persistence [7], cell differentiation [8], and the mechanics of cell wall growth [9] (Figure 1.4).

Despite the progress in imaging techniques and microfluidics, image processing remains a major bottleneck in the analysis pipelines. The unique structure of the mother machine device enables precise control of growth conditions and long-term tracking of cells, to the degree that cannot be achieved by traditional tracking of cells in microcolonies [10]. However, automated image processing is essential to process the large amounts of data generated by these high-throughput experiments. In addition, the unique structure of the mother machine device requires a specialized workflow to select and track individual growth channels. As experimentalists often need to extract precise statistics over multiple generations or observe rare events, the analysis workflow must be modular to allow inspection and curation of intermediate results.

To meet these needs, numerous mother machine-specific image analysis packages have been introduced in the last few years [11–14], in addition to general image analysis packages adaptable to the mother machine workflow [15–19]. Much recent work has been catalyzed by advances in biomedical image analysis with deep convolutional neural networks, particularly the U-Net architecture [20]. Unfortunately, many of these tools present a steep learning curve for most biologists, as they require familiarity with command line tools, programming, and image analysis methods.

Another important but underappreciated aspect of high-throughput image analysis is that researchers often use the data to derive important biological principles based on correlations. Consequently, it is important to understand the limitations and differences between different image analysis tools. Still, most labs do not have the time and resources to evaluate other tools they do not use critically, and modelers rely on experimentalists for the quality of the data in good faith.

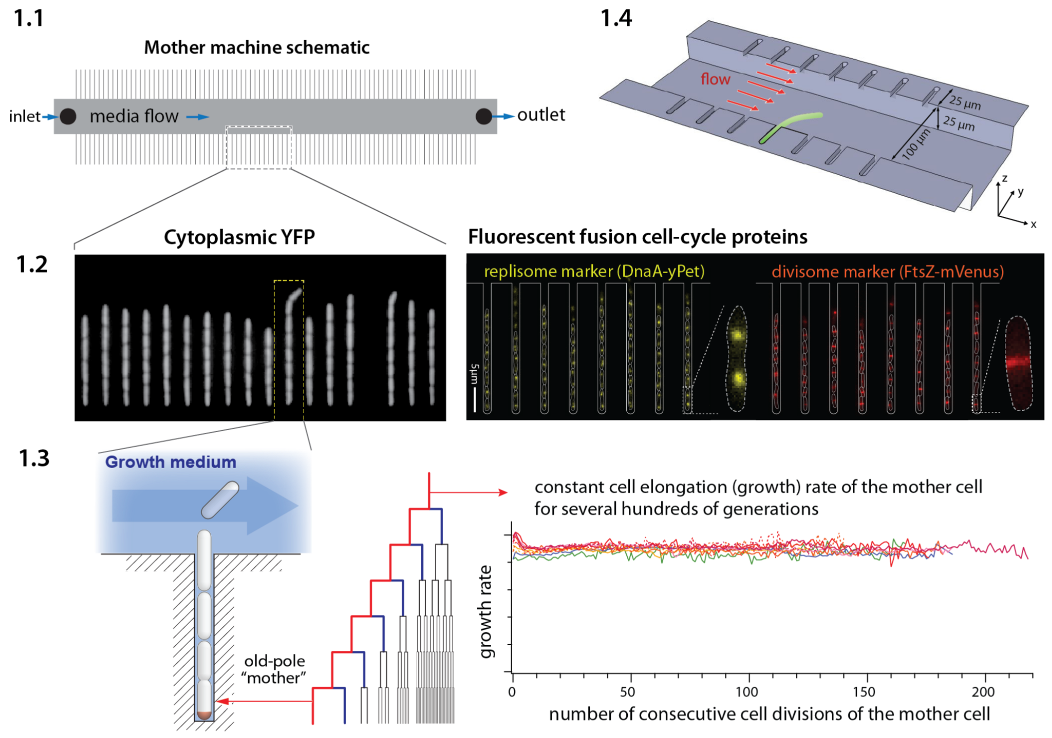

1.1
Mother machine schematic
inlet
media flow
outlet
1.4
flow
25 µm
25 µm
100 µm
z
y
x
1.2
Cytoplasmic YFP
Fluorescent fusion cell-cycle proteins
replisome marker (DnaA-yPet)
divisome marker (FtsZ-mVenus)
5µm
1.3
Growth medium
old-pole "mother"
constant cell elongation (growth) rate of the mother cell for several hundreds of generations
growth rate
0
50
100
150
200
number of consecutive cell divisions of the mother cell

Figure 1: Mother machine workflow, schematic, and applications. (1.1) Mother machine schematic. Growth channels flank a central flow cell that supplies fresh media and whisks away daughter cells. In a typical experiment, numerous fields of view (FOVs) are imaged for several hours. (1.2) Fluorescence images of *E. coli* strains expressing cytoplasmic YFP [1] (left) and markers for the replisome protein DnaN and division protein FtsZ (right) [21]. (1.3) The mother machine setup allows long-term monitoring of the old-pole mother cell lineage [1] and has other versatile applications, including (1.4) the study of the mechanical properties of bacterial cells by applying controlled Stokes forces[9].

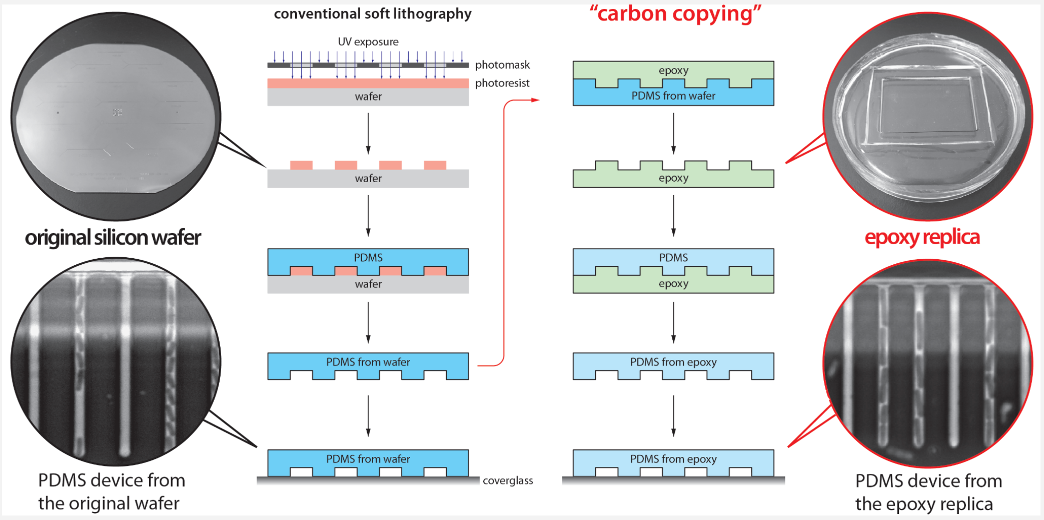
Box 1: Mother machine experimental workflow

Despite the well-appreciated power of single-cell time-lapse imaging approaches, the potential user base remains much greater than the number of researchers directly benefiting from the technology. A primary reason for this discrepancy between demand and actual adoption is the perceived cost in time and resources of investment in the required core technology: microfluidics and high throughput image analysis. Until a few years ago, setting up a typical microfluidic system for the first time took several years of training and trial-and-error, along with significant resources, for most individual labs.

Running a mother machine experiment requires the following steps: (1) fabricating a mold for the device, (2) assembling the device, (3) performing time-lapse microscopy, and (4) analyzing the images to extract time traces and statistics. To our knowledge, steps (1) and (4) have been the primary bottlenecks for most groups. Here we give a brief overview of the experimental workflow. We refer interested readers to our previous review article on single-cell physiology [22], along with other recent reviews [23,24] and published protocols [25], for a more extensive guide to single-cell imaging techniques.

Device design and fabrication. In the original mother machine design [1], narrow channels trap bacterial cells perpendicular to a larger main trench through which fresh medium flows (Figure 1.4). Several constraints apply to the design of the device. The height and width of the channels should match the dimensions of the organism under study. The channels must be large enough to facilitate the loading of the cells and allow for fast diffusion of nutrients to mother cells at the channel ends. If the channels are too deep, cells may move out of focus and potentially overlap in the z-direction, both of which impede accurate segmentation. Similarly, if channels are too wide, cells may not grow in a single file, complicating segmentation and tracking. Longer trenches will retain cells longer and allow more cells to be tracked per channel.

The prohibitive cost of mold fabrication in clean room facilities has been a bottleneck to distributing microfluidic devices. We resolved this problem using an epoxy-based fabrication technique [26], allowing us to easily and cheaply create replicative molds. Once the first microfluidic device is fabricated in the clean room, the epoxy duplication method allows us to reliably create and distribute high-fidelity device molds at a fraction of the cost of the initial fabrication. Undergraduate students in our lab routinely perform this procedure. To assist new users of the mother machine, we include a detailed procedure for the duplication method at [27].



Experiment setup. The first step of making the mother machine device is to pour PDMS (polydimethylsiloxane) onto a master mold, cure it, and remove it from the mold. Holes are punched in the cut devices at the inlet and outlet of the central channel to connect tubing for fresh medium (inlet) and waste removal (outlet) before plasma treatment (Figure 1.1). Plasma treatment covalently bonds the PDMS device to a glass cover slide or dish to be mounted on the microscope. BSA (bovine serum albumin) passed through the device passivates the surface. In our setup, we load cells to the growth channels in the device via a custom centrifuge (Figure S1). Growth medium is passed through the device using a syringe pump. The medium flow should be fast enough to clear dead cells or biofilms in the device, but slow enough that the device does not delaminate. Mounting the device on an inverted microscope requires a custom stage insert for long-term imaging. The microscope temperature must be controlled tightly.

Data analysis. Most mother machine image analysis workflows share the following steps: pre-processing the acquired images, including identification and cropping of cell traps, cell segmentation, and cell tracking. Cell segmentation is the most difficult and crucial step, as adjacent cells must be separated from each other and from device features. After accurate segmentation, the one-dimensional structure of the mother machine - which constrains the cells to move only in one direction along the length of the trap without bypassing each other - makes cell tracking relatively simple.

This article consists of three parts. First, for first-time users, we provide a brief walkthrough on implementing the mother machine in research (Box 1), including how to duplicate microfluidic devices at no cost using epoxy replicas and troubleshoot common image analysis problems. Next, we introduce MM3 [28], a fast and interactive image analysis pipeline for mother machine experiments that we have developed and used internally for over a decade. Our latest version is a Python plugin for the multidimensional image viewer napari [29]. Finally, we compare the accessibility, performance, and robustness of various current image analysis platforms. In order to trust analysis results, researchers should understand the limitations of their chosen method, especially when dealing with "black-box" approaches such as deep learning. With this in mind, we present explicit evidence that "What You Put Is What You Get" (WYPIWYG) in deep-learning-based image segmentation. Fortunately, key single-cell physiological parameter correlations and distributions are robust to the choice of analysis method, but image analysis, in general, requires caution. We provide our guiding principles and best practices to ensure transparency and reproducible results when working with mother machine data.

Results

Mother machine image analysis with napari-MM3

Analysis of time-lapse imaging experiments requires dedicated software due to the sheer volume of data produced. A typical mother-machine experiment may generate over 100GB of time-lapse image data. For instance, an experiment tracking aging might require imaging 50 fields of view (Figure 1.1) every two minutes for a week, producing a quarter of a million images. While the experimental methods for mother machine experiments have become increasingly accessible, image analysis tools have lagged. Typically, labs using the mother machine have developed their own customized analysis pipelines. Many available tools require programming experience, familiarity with command line tools, and extensive knowledge of image analysis methods. They are also often fine-tuned for specific experimental setups and difficult for the average user to adapt. Finally, existing workflows frequently require users to move between multiple interfaces such as ImageJ, MATLAB, the command line, Python scripting, and Jupyter notebooks. Newer deep learning approaches are more versatile than traditional computer vision methods. Still, they bring new issues for novices: users may need to construct their own training data and train a model, requiring a new set of tools and technical expertise, and manual annotation of training data is susceptible to human error and bias.

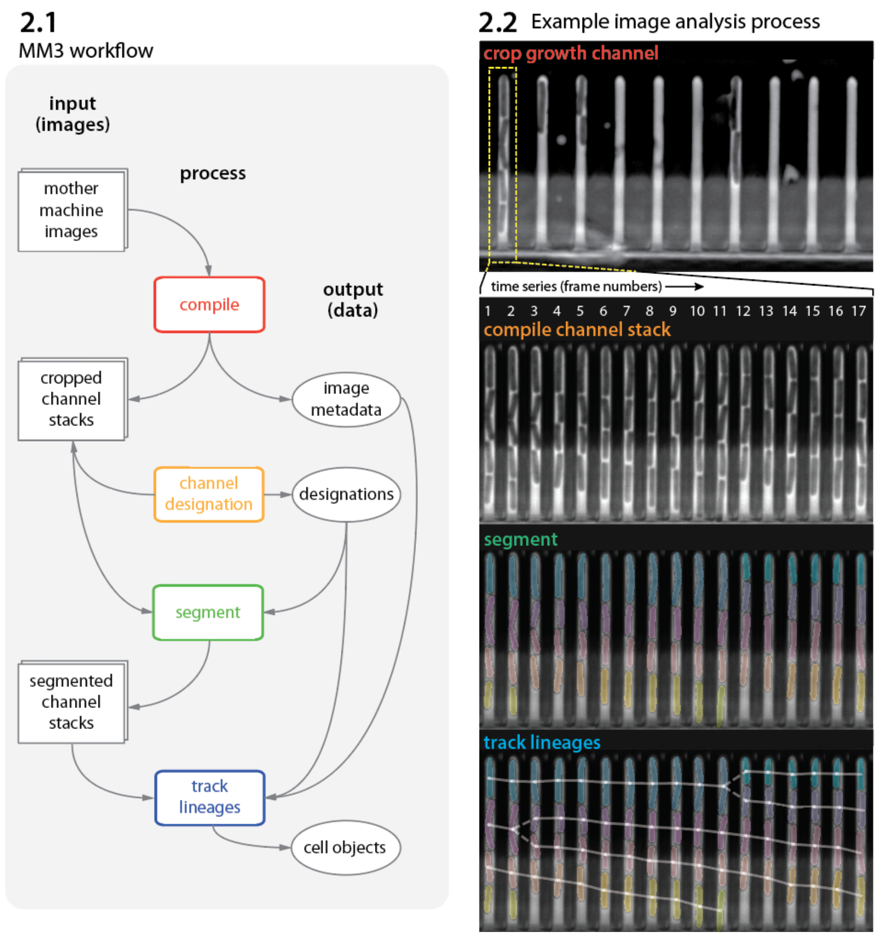


Figure 2: MM3 workflow and example images. (2.1) The MM3 image analysis pipeline takes raw mother machine images and produces cell objects. Processes (rounded rectangles) are modular; multiple methods are provided for each. (2.2) Example images from the processing of one growth channel in a single FOV. 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.

These considerations guided us in the development of our in-house analysis tool. In building MM3, we sought to provide modularity and extensive interactivity while minimizing unnecessary user intervention. MM3 aims to be a complete and flexible solution for mother machine image analysis, taking raw images and producing readily graphable cell data, while accommodating both machine learning-based and traditional computer vision techniques. It supports phase contrast and fluorescence images. It has been tested with different species (bacteria *E. coli* and *B. subtilis*, yeast *S. pombe* and *S. cerevisiae*), mother machine designs, and optical configurations. The modular pipeline architecture allows flexible use of mid-stream outputs and straightforward troubleshooting.

MM3 reflects the culmination of several iterations of our in-house mother machine analysis software developed over the past decade. Before MM3, we developed our image analysis pipeline in C++ [1] and MATLAB [30]. Eventually, Python became enormously popular, and we began MM3 as a set of Python scripts run from the command line [31]. However, the command-line-based interface had several drawbacks. The interface was more difficult for users unfamiliar with the command line or programming. It also had limited interactivity. As a result, troubleshooting was difficult and required modifying the source code to display image output at intermediate steps or manually inspecting output files in ImageJ. This made the user repeatedly move back and forth between different windows and applications, slowing the analysis.

These drawbacks motivated us to convert MM3 into a plug-in for the Python-based interactive image viewer napari [29]. napari provides an N-dimensional display ideal for visualizing multichannel time-lapse data. It offers built-in annotation tools and label layers to compare and annotate segmentation masks and tracking labels. It also provides a Python interpreter, allowing users to move easily between the viewer interface and the underlying data objects. For the best usability, we designed the napari-MM3 plug-in to allow the user to run the entire pipeline without leaving the napari interface.

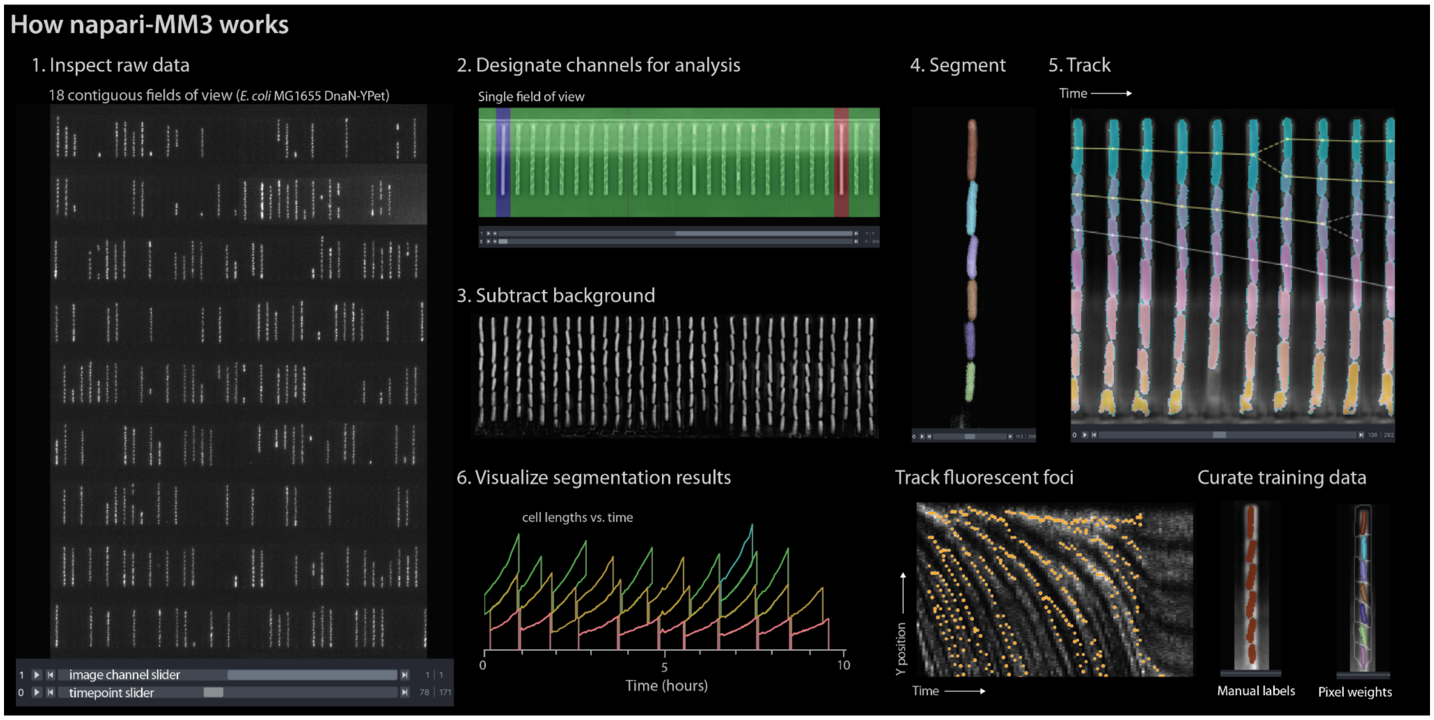


Figure 3: napari-MM3 interface. The napari viewer enables interactive analysis of mother machine data with real-time feedback and fast debugging. Raw data shown is from MG1655 background *E coli* expressing the fluorescence protein YPet fused to the replisome protein DnaN [21].

Image analysis via napari-MM3 consists of four steps (Figure 3).

1. Crop raw images and compile them into stacks corresponding to individual growth channels.
2. Choose channel stacks to be (a) analyzed, (b) used as templates for background subtraction, or (c) ignored.
3. Segment cells.
4. Construct cell lineages. napari-MM3 treats individual cells in the lineages as objects that can be plotted directly or converted to another data format.

We elaborate on these steps as follows.

1. Channel detection and curation

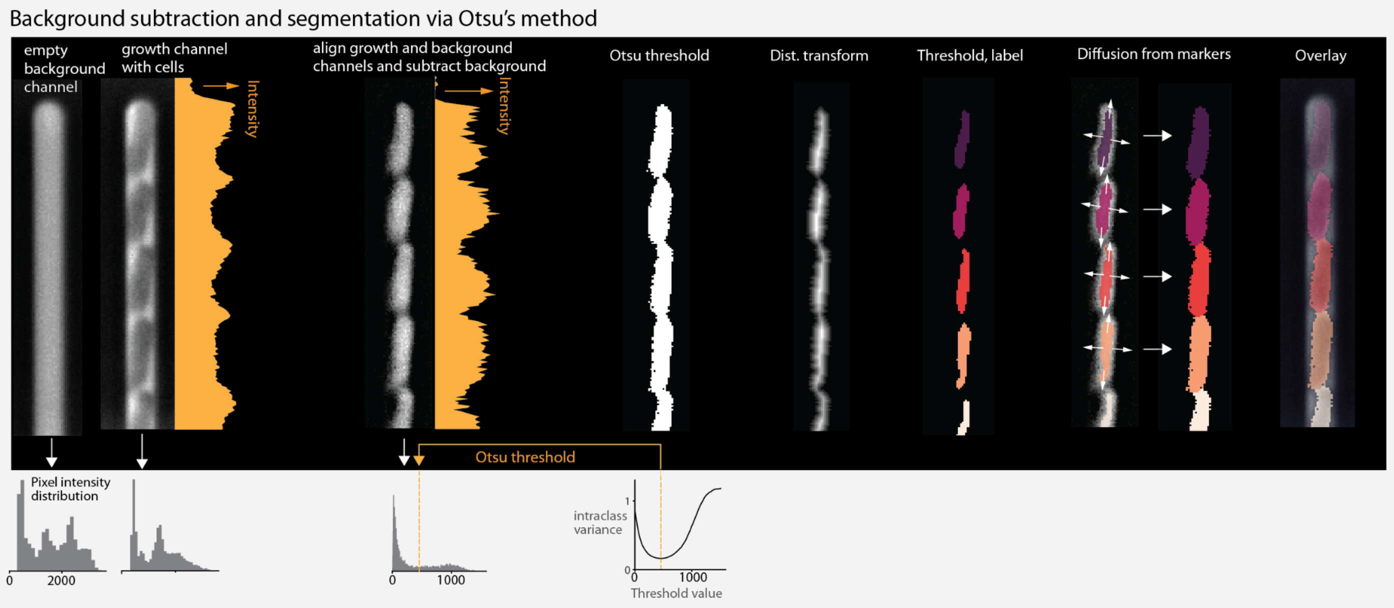
The first section of the napari-MM3 pipeline takes in raw micrographs and returns image stacks corresponding to one growth channel through time. napari-MM3 detects channels using a wavelet transform and then aligns them over time to correct for stage drift and vibration. The aligned growth channels are saved as unique image stacks with all time points for a given growth channel and color channel. As not all growth channels contain cells, and napari-MM3 auto-detects channels as full or empty based on the time correlation of the y-profile of the growth channel. The auto-detected growth channels and their classifications are then displayed in the napari viewer for the user to inspect and modify as needed.

2. Cell segmentation

napari-MM3 offers two methods for cell segmentation, one using traditional computer vision techniques and the other using deep learning. The non-learning method utilizes Otsu's method to apply a binary threshold to separate cell objects from the background. It then labels the isolated cells and uses a watershed diffusion algorithm to fill out the cell boundaries. This method is fast but optimized for specific mother machine designs and phase contrast imaging of bacteria. It also requires accurate background subtraction of phase contrast images (Box 2), to ensure that the presence of the channel border does not interfere with cell detection. The supervised learning method uses a convolutional neural net (CNN) with the U-Net architecture [20]. The napari viewer can be used to construct training data, with the option to import existing Otsu or U-Net segmentation output as a template. The neural net can then be trained directly from napari, with the option to check the performance of the model in the napari viewer after successive rounds of training.

Box 2: Segmentation via Otsu's method

The Otsu segmentation method first aligns the growth channel of interest with an empty background channel by computing the orientation, which maximizes the pixel-wise cross-correlation. The empty channel is then subtracted from the full channel, and the image is inverted. This background subtraction step is essential, as it removes the dark image of the PDMS device, which will otherwise interfere with segmenting the (dark) cells. Otsu's method [32] is applied to find the binary threshold value, which maximizes the inter-region variance. We then apply a Euclidean distance transform, wherein each pixel is labeled with its distance to the dark region. The image is thresholded again, and a morphological opening is applied to erode links between regions. Small objects and objects touching the image border are removed. Each region is labeled, and the labels are used to seed a random walker algorithm [33] on the original image.



3. Cell tracking and lineage reconstruction

Finally, napari-MM3 links segmented cells in time to define a lineage of cell objects, using a simple decision tree based on a priori knowledge of binary fission and the mother machine. Tracking produces a dictionary of cell objects containing relevant information derived from the cell segments, including the cell lengths and volumes over time, cell elongation rate, and generation time. Plotting can then be done directly from the cell objects, or the cell objects can be converted into a .csv file, a pandas DataFrame, or a MATLAB structure. We provide a Jupyter notebook demonstrating this analysis at [34].

Additional features

napari-MM3 also offers several additional modules supplemental to the main processing pipeline, including methods for fluorescence image analysis and U-Net training data construction and model training. Integrated fluorescence signal and fluorescence per cell area and volume for each timepoint can be extracted using the "Colors" module. napari-MM3 also includes a module for the detection and tracking of fluorescent spots or "foci." For example, we have used it to track fluorescently labeled replisome machinery in bacteria in order to measure the timing and synchrony of DNA replication initiation [2,21]. Lastly, U-Net segmentation training data can be constructed by manual annotation of raw images in the napari viewer. napari-MM3 offers the option to construct training data with existing Otsu or U-Net segmentation data as a template. This allows the user to iteratively train a model, correct mistakes in its output, and use the modified output as input for the next round of training. We also provide a Jupyter notebook covering training data construction and model training at [34].

	Channel detection	Background subtraction	Segmentation (Otsu)	Segmentation (U-Net)	Tracking	Total (Otsu)	Total (U-Net)
Frame processing time	N/A	2 ms	4 ms	5.3 ms	N/A	N/A	N/A
Channel stack processing time (262 time frames)	N/A	0.54 sec	1.14 sec	1.4 sec	0.7 sec	3.1 sec	2.1 sec
FOV processing time (35 channels)	14.1 sec	17.5 sec	36.5 sec	46 sec	46.7 sec	2 min	1.7 min
Exp. processing time (26 GB, 34 FOVs, ~20,000 cells)	3.2 min	9.9 min	20.6 min	26 min	26.4 min	60 min	55 min

Table 1: Performance metrics for napari-MM3. Processing times were measured on an iMac with a 3.6 GHz 10-Core Intel Core i9 processor with 64 GB of RAM and an AMD Radeon Pro 5500 XT 8 GB GPU. The dataset analyzed is from [21] and consists of 26 GB of raw image data (12 hours, 262 time frames, 2 imaging planes, 34 FOVs, and ~35 growth channels per FOV). Note that while the Otsu segmentation method is slightly faster than the U-Net, it also requires a background subtraction step, such that the total runtimes of the two methods are comparable.

Performance test of napari-MM3

To evaluate the speed of napari-MM3, we timed the processing of a typical dataset (Table 1). Using consumer-grade hardware, a single-channel stack consisting of several hundred time frames can be processed in less than five seconds, and a typical experiment consisting of 25 GB of imaging data can be processed in under an hour. These metrics are on par with those reported by other recently published mother machine software [14,35,36].

Software	**Implementation**	**Segmentation**	**Deep learning support**
BACMMAN [14] / DistNet [39]	ImageJ plugin	2D	✓
DeLTA [11,36]	Python package	2D	✓
napari-MM3 [31], this work	napari plug-in	2D	✓
SAM [35]	MATLAB	2D	
MMHelper [13]	ImageJ plugin	2D	
MolySo [12]	Python package	1D	
MoMa [37]	ImageJ plugin	1D	

Table 2: Overview of mother-machine image analysis tools. A comparison of several published imaging methods. '2D' or '1D' segmentation indicates whether the cells are labeled in an image and analyzed in two dimensions, or projected onto a vertical axis and analyzed in one dimension. Several tools support the use of deep learning (as compared to classical computer vision techniques, such as Otsu's method).

Comparison with other image analysis software

We also tested napari-MM3's usability and performance against other popular software. We began by surveying a range of existing mother machine image analysis tools (Table 2). Some early analysis pipelines used one-dimensional segmentation methods [12,37], which perform adequately when cells are tightly confined in the growth channels. In recent years, many excellent general-purpose CNN-based robust cell segmentation tools have been developed [15–18,38], which may be extended to process mother machine data.

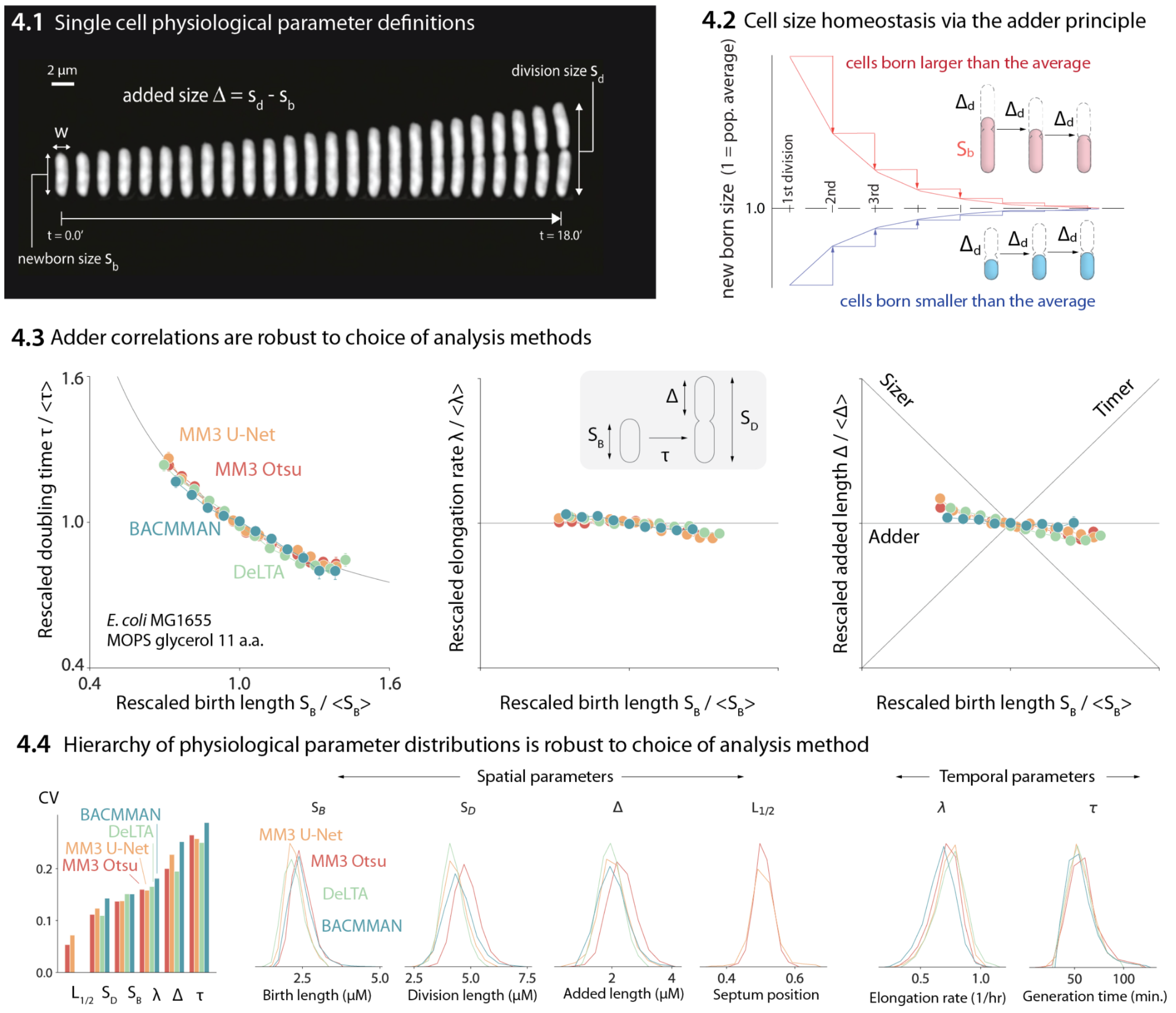


Figure 4: Comparison of various image analysis approaches. (4.1) A time series of a typical cell growing in a nutrient-rich medium. The birth size, division size, and added size are indicated. (4.2) The adder principle ensures cell size homeostasis via passive convergence of cell size to the population mean. (4.3) We analyzed multiple datasets from our lab using MM3, DeLTA, and BACMMAN, and obtained robust correlations between birth length, doubling time, elongation rate, and added length. Representative results from one dataset [21] for MG1655 background *E. coli* grown in MOPS glycerol + 11 amino acids are shown, with 9,000 - 13,000 cells analyzed depending on the method. (4.4) Distributions of key physiological parameters are independent of the analysis methods. The data and code used to generate this figure are available at [40]

In this work, we only tested mother-machine-specific pipelines. In particular, we constrained our analysis to DeLTA and BACMMAN, two excellent open-source mother machine-specific pipelines offering 2D segmentation and cell tracking, which are also well-documented and actively maintained. BACMMAN [14] performs 2D segmentation via traditional computer vision methods similar to those implemented in napari-MM3 and has recently added support for CNN-based segmentation as well [41]. DeLTA [11,36] uses the U-Net architecture for channel detection, cell segmentation, and cell tracking, with a mother machine-specific and general agar pad mode. We used BACMMAN, DeLTA, and napari-MM3 to analyze the same published dataset [21] consisting of *E. coli* MG1655 grown in minimal growth medium (MOPS 0.4% glycerol 11 amino acids with ~60-minute doubling time) [21]. Data processed in napari-MM3 was separately segmented with U-Net and traditional computer vision (Otsu thresholding) methods. We found that the pre-trained mother machine model provided with DeLTA did not generalize well to our data. However, after training a new model with representative data, we achieved accurate segmentation.

We compared the distributions and correlations of key physiological parameters generated by each analysis tool, motivated by our standard approach to single-cell physiology [5,21,30,42]. First, we confirmed that all four analysis methods yield essentially identical correlations between cell length at birth (S_B) vs. (a) generation time (τ), (b) elongation rate (λ), and (c) the length added between birth and division (Δ) (Figure 4.3). Next, we compared the distributions of various physiological parameters. The CV (coefficient of variation) of a physiological parameter distribution is often taken to reflect the tightness of the underlying biological control. We have previously found [2,30] that the CVs of a set of physiological parameters (birth length, division length, length added between divisions, growth rate, generation time, and septum position) are invariant across growth conditions in *E. coli* and *B. subtilis*, and that the hierarchy of CVs is preserved across the two evolutionarily divergent species [2,30]. Here, we confirmed that the distributions of these physiological parameters are independent of the analysis methods (Figure 4.4). In particular, the hierarchy of CVs is preserved by all three methods tested. Last, while in this dataset the old-pole "mother" cells showed signs of aging (in particular, a reduced elongation rate), we note that this aging phenotype is strain- and condition-dependent (Figure S3).

Systematic discrepancies in cell segmentation propagate to deep learning predictions

While we found that the correlations between physiological parameters were preserved across the different analysis methods (Figure 4.3), we also observed systematic discrepancies in the results obtained by different methods, including cell length at birth (S_b), length at division (S_d), and length added between birth and division (Δ) (Figure 4.4). In particular, napari-MM3's Otsu segmentation method systematically generated larger cell masks than napari-MM3 U-Net, DeLTA, and BACMMAN (Figure 4.4). We focused on the discrepancies between the two MM3 outputs. Although the deviation between the two masks may not appear significant when individual masks are inspected by eye (Figure 5.1), the Otsu method yields cells that are 5%-10% larger at each time point than those returned by the U-Net method when averaging over an entire experiment with tens of thousands of cells tracked (Figure 5.2).

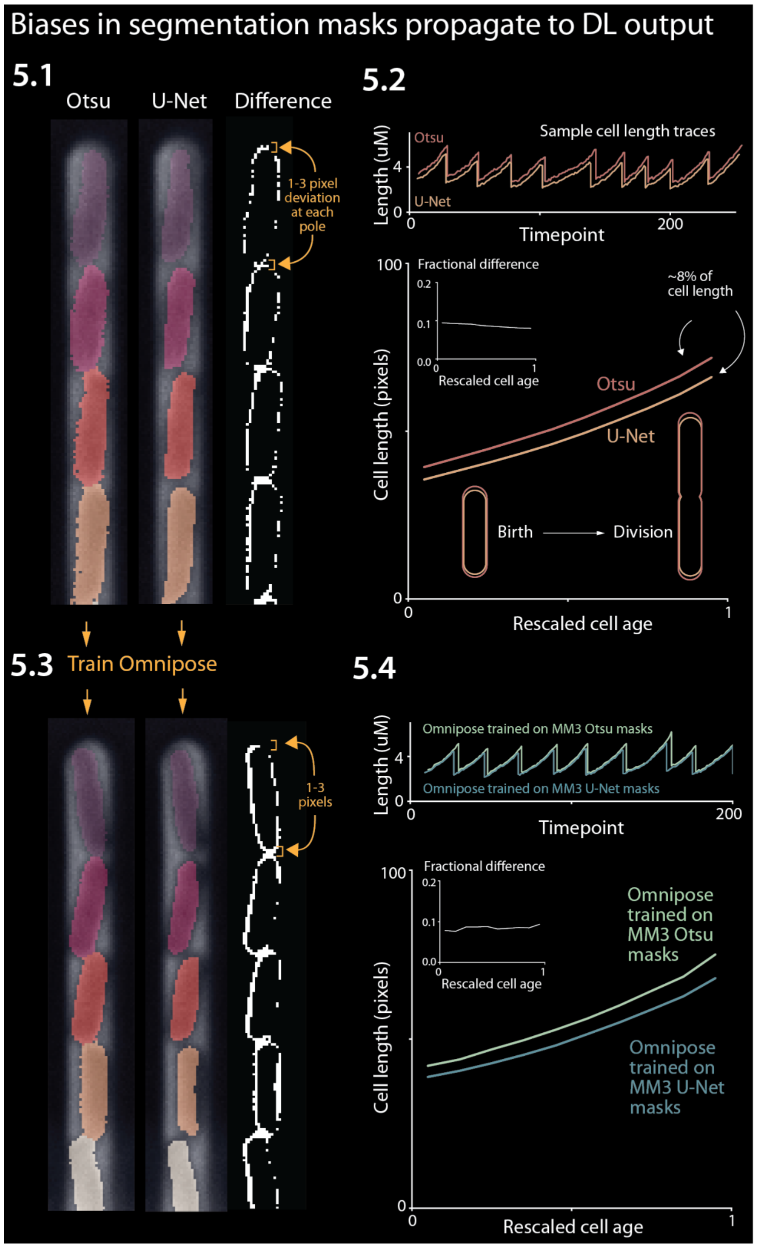


Figure 5: Effect of systematic deviation in segmentation output from Otsu and U-Net methods. 5.1. Otsu and U-Net segmentation masks. The Otsu method systematically yields masks that are 5%-10% larger than the other methods. 5.2. We confirmed that this discrepancy occurs consistently across the cell cycle. 5.3 We trained the Omnipose model on masks generated by either napari-MM3 Otsu or napari-MM3 U-Net separately. 5.4. The systematic discrepancy in the training data masks propagated to the output of the trained models, confirming WYPIWYG.

We found that the Otsu and U-Net methods had similar accuracy in identifying cells, with both closely matching the user-annotated ground-truth masks (Figure S2). Therefore, the discrepancy does not arise from one method yielding erroneous output. Rather, in our view, it is a consequence of the difficulty of consistently distinguishing cell boundaries by eye, and the sensitivity of segmentation outputs to threshold values. In classical segmentation methods, user-set threshold values can systematically alter the final output cell size. Indeed, the two classical methods tested here - MM3 Otsu and BACMMAN's non-learning method (which also uses Otsu thresholding) - output different cell masks with their 'default' parameter settings. In addition, binary U-Net segmentation methods, such as those implemented in napari-MM3 and DeLTA, output smaller cell sizes because the model must leave a gap between cells so that they are not stitched together (note this is not a fundamental limitation of U-Net, but a consequence of our implementation: see, e.g. [17] or [39] for more complex approaches which avoid this issue).

The danger of WYPIWYG (What You Put Is What You Get) in deep-learning-based image analysis

Given that classical methods are clearly sensitive to this threshold tuning, we wondered whether deep-learning approaches would also be as sensitive to their inputs or to user bias [43,44]. We found that is indeed the case. We chose the recent cutting-edge segmentation model Omnipose [17] and separately trained it on masks derived from the aforementioned Otsu segmentation output and masks from the napari-MM3 U-Net segmentation output. We chose Omnipose as it assigns different labels to different cells, and can thus segment cells with contiguous boundaries, in contrast to MM3 or DeLTA's U-Net implementations. Strikingly, we found that the systematic discrepancy in the training masks propagated to the output of the trained models: the Omnipose model trained on larger Otsu masks generated larger masks upon evaluation with the same data, while the Omnipose model trained on smaller U-Net masks output smaller masks (Figure 5.3), explicitly confirming what we call WYPIWYG. That is, at least for our setup, systematic differences in training data masks lead the model to learn different threshold intensity values and thus to systematically output larger or smaller masks. We emphasize this result does not reflect a flaw in Omnipose - whose performance we found impressive - but rather a limitation of deep learning methods in general [43].

Discussion

In this study, we introduced a modular and interactive image analysis pipeline for mother machine experiments, and compared its effectiveness to other existing tools. Unlike its predecessors, napari-MM3 is equipped with an intuitive and modular interface, making it highly accessible to new users. Our main goal is to lower the barrier to entry in image analysis, which has been a primary obstacle in adopting the mother machine, and ultimately increase its user base.

Finally, we discuss common challenges faced by users new to high-throughput image analysis and give our prescriptions for overcoming them.

Validating results

We showed that distributions and correlations in key cell cycle parameters are invariant to the choice of analysis pipeline, provided that care is taken in parameter adjustment and postprocessing. However, this parallel processing of data is not feasible for every experiment. Instead, we suggest users can validate their results in the following ways:

1. A qualitative "eye test" is an important first step: one should always visually inspect one's data. Often, this may be sufficient to establish whether the analysis is operating as expected.
2. When a more quantitative and systematic approach is needed, the user can compare the output of their analysis to a subset of manually annotated 'ground truth' images. Quantitative measures such as the Jaccard index [44] or dice index may be used. These metrics are particularly useful for comparing the results of different parameter choices in a given method, allowing the user to determine the combination that yields the most accurate segmentation or tracking results.
3. Verify that the averages calculated from single-cell measurements match the results of population-level control experiments.
4. When possible, filter for subsets of the data that are likely to reflect accurate segmentation and continuous tracking, such as cell lineages that are continuously tracked for the duration of the experiment.

Choosing an image analysis tool

For many years, published and well-documented pipelines for mother machine image analysis were scarce, and existing software required extensive parameter reconfiguration, knowledge of image processing techniques, and programming experience to use effectively. In recent years, advances in deep learning have contributed to a rapidly growing set of image analysis tools that perform cell segmentation and tracking.

For users trying to choose the appropriate tool for their data, we suggest the following points to keep in mind:

1. Tools that are actively maintained, with an easy way to contact the developer, will be more likely to work well and will be easier to troubleshoot than others.
2. Detailed documentation and tutorials are valuable, and will allow the user to troubleshoot the software without direct guidance from the developers.
3. Depending on the user's level of comfort with coding, it may be beneficial to choose a tool that is implemented through a graphical user interface and does not require additional programming. Moreover, even for programmers, we found within our lab that introducing interactivity when necessary dramatically expedited the data analysis process.
4. Full stack (vertically integrated) tools that cover the entire analysis pipeline may save time and work, relative to those which only perform a portion of the needed analysis.
5. It is worthwhile to engage with the online community around the tool, if one exists. We have found the image.sc forum [45] valuable in the past, in particular for help with napari.
6. Consider whether the tool is open source or requires a license. With regard to this point, we encourage tool developers to avoid proprietary software such as MATLAB, which may not be accessible to all users. The open-source Java-based image-processing program ImageJ [46] has been a dominant tool in biological image analysis for many years. The recent growth of image analysis and machine learning tools in Python makes napari [29] an attractive alternative to ImageJ.

Traditional computer vision vs. deep learning methods

A key choice many users will face is whether to use deep learning-based or traditional methods for image analysis. The field has increasingly shifted toward deep learning methods, and this shift will likely accelerate. While traditional computer vision methods remain useful, deep learning-based methods have a clear advantage in their ability to generalize quickly to new datasets.

In our lab, we have found that traditional computer vision techniques perform excellently on cell segmentation and tracking in the mother machine, subject to constraints on the experimental setup. However, such methods often require extensive reconfiguration or fail entirely when applied to data obtained under new biological conditions (different organisms, different cell morphology) and imaging conditions (varied illumination, microscope setup). Our own non-learning segmentation method performs well, provided that cells are tightly confined in the mother machine channels and do not move substantially. Prior to the adoption of deep learning methods, this requirement necessitated the design of different devices for cells grown in different growth conditions, as the cell width in some *E. coli* strain backgrounds varies with the population growth rate.

By contrast, the key strength of deep learning approaches is their ability to generalize to new conditions - whether to different illumination conditions, different types of input images (phase contrast, brightfield, fluorescence) or different organisms and cell types entirely. The main barrier to adoption of learning-based methods remains the construction of training data, which can be tedious and time-consuming. A training data set of 50 - 100 images comprising several hundred cells can be constructed in a few hours and will achieve passable segmentation on representative data. However, larger training sets on the order of thousands of images are preferable, and will yield improved model accuracy and generalizability. The time needed for annotation can be reduced by seeding the data with masks generated by classical methods - or iteratively seeding with U-Net output - and then refining the masks further by hand. Model performance and generalizability can often be significantly improved by augmenting training data via manipulations such as rotating or shearing, distorting the intensity profile, and adding noise. Nonetheless, we have found that even with extensive data augmentation, applying the U-Net segmentation to new experimental configurations or imaging conditions often requires retraining the model on an expanded dataset with more representative data. Ultimately, deep learning methods are only as good as the data they are trained on, and are most likely to fail when training data is insufficient, mislabeled, or not representative. Going forward, sharing of training sets and models [47] between different groups can facilitate progress and aid reproducibility.

In addition to deep learning-based segmentation, learning-based cell tracking in the mother machine has been implemented recently by multiple groups [11,39]. For cells growing unconstrained on 2D surfaces such as agar pads, U-Net tracking dramatically outperforms traditional methods [11]. However, for steady-state growth in the mother machine where cells are confined and constrained to move in one dimension, we have not found a significant difference between the performance of deep learning-based tracking and the non-learning tracking method implemented in MM3. In both cases, errors in tracking nearly always arise from errors in segmentation. However, deep learning-based tracking may offer an advantage in cases where cells may move substantially along the length of the channel, or undergo dramatic morphological changes such as filamentation.

Ultimately, for groups with existing analysis pipelines fine-tuned for specific organisms under specific imaging conditions to perform simple tasks such as segmentation and 1D tracking, there may be little incentive to switch to deep learning methods. However, for users looking to develop a new pipeline or analyze more complex data, the power and generality of deep learning tools will make them the method of choice.

Should you worry about WYPIWYG and systematic discrepancy in segmentation results between different methods?

Given the 5%-10% variance in the segmented bacterial cell size is comparable to the CVs of several physiological parameters (Figure 4), should researchers be concerned about the robustness of their results? The answer depends on the purpose of the image analysis.

If the research critically relies on the absolute cell size, such as cell-size control [21,30], the researcher must be aware of inherent limitations to the accuracy of spatial measurements from cell segmentation. These arise in part from the difficulty of consistently distinguishing cell boundaries by eye. Once a threshold is chosen, the choice will affect all analyzed cells systematically. This limitation applies to both deep learning (through the construction of training data) and traditional computer vision methods (through the manual input of a threshold value). For cell segmentation, the uncertainties are typically comparable to the pixel size of the images, rather than optical resolutions. For example, the pixel size in the images in Figure 5 is 0.065 μm (for the camera pixel size 6.5 μm and 100X magnification), which is non-negligible for many commonly cultured bacterial cells with submicron cell widths - e.g., *Enterobacterales*, *Pseudomonas*, *Bacillus subtilis,* and *Caulobacter crescentus*. For most commercially available cameras and objective lenses used in quantitative bacterial cell biology, 10% should be taken as a conservative lower bound for uncertainty when comparing absolute spatial measurements of bacterial cell size.

Indeed, researchers should be particularly careful when comparing absolute measurements of cell size, e.g., at division or initiation of chromosome replication obtained by different groups using different image analysis methods. While absolute temporal measurements are more robust than spatial measurements (Figure 4.4), the differences in spatial measurements can propagate to the measured timing of, e.g., cell division. For instance, we observed that the Otsu method stitched cells together for slightly longer than the U-Net method did (Figure 5.2), but as this shift applied equally to birth and division, it did not affect the average cell generation time (Figure 4.4).

Fortunately, the examples mentioned above are extreme cases. For instance, the pixel-size uncertainties will reflect a smaller proportion of the cell size when imaging larger cells such as yeast or mammalian cells. Even in our research on single-cell bacterial physiology [2,21,30], we find that correlations and relative changes are more likely to be robust than absolute spatial measurements to the choice of analysis method (Figure 4). Furthermore, different applications of deep-learning based image analysis, such as high-throughput phenotypic classification [48] will be much more robust to the pixel-size uncertainties in image segmentation results.

In recent years researchers have extracted increasingly complex information from mother machine data, much of which is beyond the scope of our work. Ultimately, researchers should test the robustness and reproducibility of their novel analysis. To aid those interested, we have made the raw data analyzed here publicly available [49]. We encourage other researchers to perform their own analyses and directly compare the results.

Conclusion and recommendations

Here, we presented a guide to first-time users of the mother machine, introduced our updated image analysis software, and validated it against existing tools. In contrast to existing tools, napari-MM3 provides a simple and modular user-friendly interface, which we believe makes it uniquely accessible and valuable to novice users. By lowering the barrier to entry in image analysis - the key bottleneck in mother machine adoption - we aim to increase the user base of this powerful tool dramatically.

After testing two other well-constructed mother machine image analysis pipelines, we concluded that all four methods (BACMMAN, DeLTA, MM3 Otsu & MM3 U-Net) yielded consistent and reproducible results. Thus, for users already comfortable with a given pipeline, there is no strong incentive to switch to a new one. However, the different pipelines do have markedly different user interfaces. DeLTA is set up to provide a simple "one-shot" analysis, in which image preprocessing, channel detection, segmentation, and tracking are performed in sequence with minimal user input. This arrangement simplifies the analysis process, especially for first-time users. In particular, it can be helpful for users who want to quickly verify that the software will serve their purpose, before investing more time in setting up and running the analysis. On the other hand, the intermediate steps in the pipeline are less accessible, which may make debugging and troubleshooting more involved. BACMMAN, like napari-MM3, is more modular than DeLTA. This modularity can aid troubleshooting, but first-time users may also be overwhelmed by the sheer number of configurable parameters and steps. With napari-MM3, we attempted to strike a balance between these two well-designed and well-performing tools, while taking advantage of the fast-growing next-generation image analysis platform napari. napari-MM3 attempts to infer or pre-set as many parameters as possible, while the napari interface makes midstream output easily accessible. We have been using MM3, and more recently napari-MM3, for over a decade since our introduction of the mother machine in 2010, and we will continue to actively maintain and improve it in the coming years.

The mother machine setup has become increasingly accessible to researchers in recent years, through the distribution of molds and the publication of in-depth protocols and open-source image analysis software. At the same time, new variations of the device have found diverse applications, including bacterial starvation [6], mammalian cell growth [50,51], and genetic screening [52,53]. Clearly, the combination of microfluidics with high-resolution time-lapse imaging remains powerful among single-cell techniques. We hope that this article will prove useful to mother machine veterans and first-time users alike.

Acknowledgements

This work was supported by the Chan Zuckerberg Initiative grant NPA-0000000033, NIH grant R35GM139622 and NSF grant MCB-2016090.

We thank Mara Casebeer, Thias Boesen, and the members of the Jun Lab for testing and debugging napari-MM3. We also thank Kevin Cutler for helping us to install Omnipose and run it on our data.

Methods

Getting started with napari-MM3

napari-MM3 is implemented entirely in Python and can be accessed at [28], along with documentation covering installation and usage. It will run on a standard Mac, PC, or Linux machine. We recommend using the Anaconda Python distribution to simplify installation. Furthermore:

- A notebook providing functions for postprocessing and plotting of the napari-MM3 output is available at [34].
- A video tutorial walkthrough for napari-MM3 is available at [54].
- The processed data analyzed in this manuscript is available at [55].

Mother machine protocols

The protocols for epoxy mold fabrication and duplication of the PDMS devices are available on our lab's Github [56].

Imaging conditions

The data analyzed in Figures 4 and 5 (originally published in [21]) was obtained on an inverted microscope (Nikon Ti-E) with Perfect Focus 3 (PFS3), 100x oil immersion objective (PH3, numerical aperture = 1.45), and Obis laser 488LX (Coherent Inc., CA) as a fluorescence light source, and an Andor NEO sCMOS (Andor Technology) camera. The laser power was 18 mW. The exposure time was 200 ms for phase contrast imaging and 50 ms for fluorescence.

Image analysis for software comparison

For the software comparison in Figure 4, we analyzed a dataset from [21] consisting of *E. coli* MG1655 expressing a fluorescent protein YPet fused to the replisome protein DnaN. The cells were grown in MOPS minimal medium + glycerol and 11 amino acids. The dataset was analyzed end-to-end starting from the raw .nd2 file with BACMMAN, DeLTA, and MM3. For analysis with DeLTA, we used the provided channel detection and tracking models but trained a new model on our own data for segmentation. For segmentation with BACMMAN, we used the standard non-learning phase contrast segmentation method 'MicrochannelPhase2D'. Postprocessing of the output of each pipeline was done in Python. For each pipeline, we filtered for cells whose mothers and daughters were also tracked.

The code and data to reproduce the plots in Figure 4 are available at [34] and [57], respectively.

For the comparison of Otsu and U-Net outputs from Omnipose in Figure 5, we trained Omnipose with a learning rate of .01 without a pre-trained model. We used the same set of 1000 randomly selected images for both Otsu and U-Net, the only difference coming from the labeled masks themselves. Both models were trained until the loss dipped below 0.9 (390 epochs for U-Net, 210 epochs for Otsu). In some cases, the model "hallucinated" cells along the channel features. We excluded these images from the final analysis.

Overview of the MM3 pipeline

Channel compilation and designation

The first section of the MM3 pipeline takes in raw micrographs and returns image stacks corresponding to one growth channel over 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 is performed using a wavelet transform, in which a mask is made which is applied across all time points. Channels are cropped through time using the masks and saved as unique image stacks that include all time points 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. Consequently, MM3 auto-detects empty and full channels based on the time correlation of the y-profile of the channel (empty channels are highly correlated in time, while channels containing cells are not). The autodetected channels and their classifications are then displayed in the napari viewer for the user to inspect and modify as needed. The user may also manually select empty channels free of artifacts to be used as templates for phase or fluorescence background subtraction.

Background subtraction

MM3's Otsu segmentation method requires background subtraction of phase contrast images. The subtraction ensures that the presence of the channel border does not interfere with detection of cells. To this end, we overlay the previously-identified empty channels on the full channels to be subtracted. The two channels are aligned such that the cross-correlation of overlaid pixels is maximized. After the inversion of the image, this leaves the cells as the only bright objects on a dark background. Good alignment of the device features in the empty and full channel is essential here. Imperfect alignment will leave artifacts in the subtracted image, which interfere with later steps, and is a common failure point for this method. Note that the subtraction step necessitates the presence of some empty channels in each experiment. The U-Net segmentation does not require background subtraction.

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, specifically background subtraction, Otsu thresholding, morphological operations, and watershed algorithms. As the standard method may require fine-tuning of parameters, the napari plugin allows the user to quickly preview the effect of tuning morphological parameters and threshold value on the segmentation output, without having to process the entire dataset. The Otsu segmentation method first aligns the channel of interest with an empty background channel by computing the orientation, which maximizes the pixel-wise cross-correlation. The empty channel is then subtracted from the full channel, and the image is inverted. Otsu's method is then applied to find the binary threshold value which maximizes the inter-region variance (or equivalently, minimizes the intra-region variance). We then apply a Euclidean distance transform, in which each pixel is labeled with its distance to the dark region. The image is thresholded again, and a morphological opening is applied to erode links between regions. Small objects and objects touching the image border are removed. Each region is labeled, and the labels are used to seed a random walker algorithm [33] on the original image.

The supervised learning method uses a standard U-Net architecture with five levels [20]. The napari viewer can be used to construct training data, with the option to import existing Otsu or U-Net segmentation output as a template. The neural net can then be trained using a separate widget, with the option to check the performance of the model in the napari viewer after successive rounds of training. We found that applying a weighted loss depending on pixel location - as suggested in the original U-Net paper [20] and implemented in DeLTA [36] - sped up model training and improved segmentation and tracking. Since the accurate separation of adjacent cells is vital for cell tracking, the cost of misidentifying pixels between bordering cells is high. We initially implemented a simple binary weight map where pixels between cells were weighted highly and all others pixels relatively lower. We later added a more complex mapping, drawing directly from the one implemented in DeLTA [11], where weights are maximized on the skeletons [58] of the cells and borders. Intuitively, this weighting tells the model that pixels in the center of the cell, in regions far from cells, and on the borders between cells are most important to predict accurately.

Illumination conditions can vary across laboratories, microbial species, and with device design. To aid the generalizability of the U-Net model, on specific conditions, we augmented the training data with various morphological techniques, including changing magnification, zooming and rotating, and Gaussian noise and blur. We also adapted several non-standard operations from DeLTA, one which performs elastic deformation and two others that distort image contrast to simulate changes in illumination within the field of view and between experiments.

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. The default tracking method 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 lineage tree obtained by tracking is displayed in the napari viewer in the form of a kymograph, in which the x-axis represents time, and cell linkages and divisions are indicated by forking lines.

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 that 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 modified and extended by 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.

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. We provide a Jupyter notebook [34] to illustrate how the data can be extracted and plotted.

Fluorescence analysis

Integrated fluorescence signal and fluorescence per cell area and volume for each timepoint can be extracted using the Colors module.

Focus tracking

The focus tracking module enables the identification and tracking of fluorescent spots or ‘foci.’ This module has been used in our lab for tracking fluorescently labeled replisome machinery in bacteria in order to measure the timing and synchrony of DNA replication initiation. However, it may be applied to any use case requiring localization and tracking of intracellular spots. The module uses a Laplacian convolution to identify fluorescent spots. Foci are linked to the cell objects in which they appear.

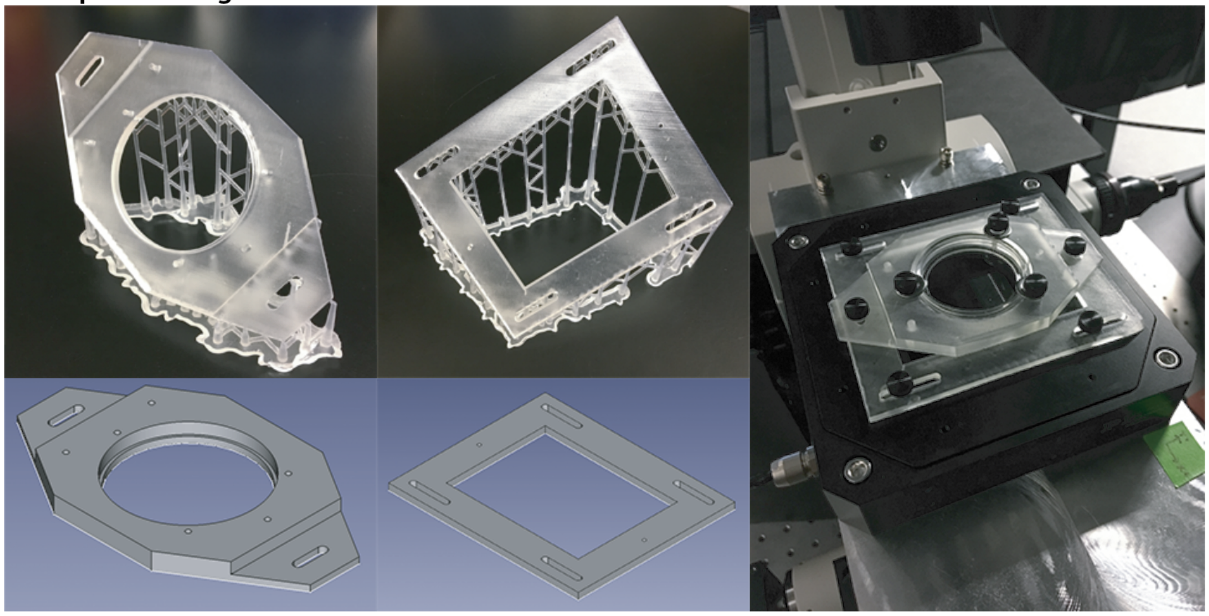
U-Net training data annotation and model training

Training data can be constructed by manual annotation of raw images in the napari viewer. MM3 offers the option to construct training data with existing (Otsu or U-Net) segmentation data as a template. This allows the user to iteratively train a model, correct mistakes in its output, and use the modified output as input for the next round of training.

Training data was augmented as described above to aid the generalizability of the model. The U-Net model was trained over 500 epochs with a batch size of 8 samples. We used a training-validation split of 90-10 to reduce the risk of overfitting.

Figure S1. Inexpensive fabrication of stage insert and cell loader with 3D printing.

A. 3D printed stage insert



B. 3D printed cell loader

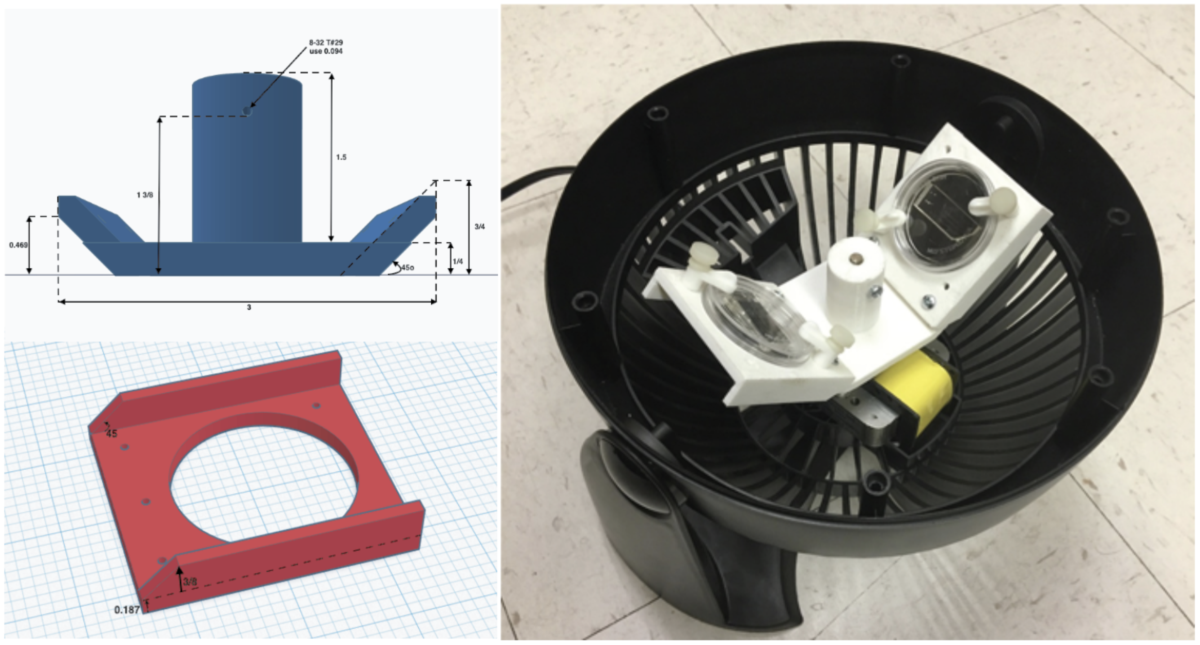


Figure S2: Segmentation accuracy of napari-MM3 Otsu and U-Net methods

To quantify the accuracy of the segmentation masks generated by MM3's Otsu and U-Net segmentation methods, we computed the Jaccard Index [44,59] as a function of the intersection-over-union (IoU) threshold.

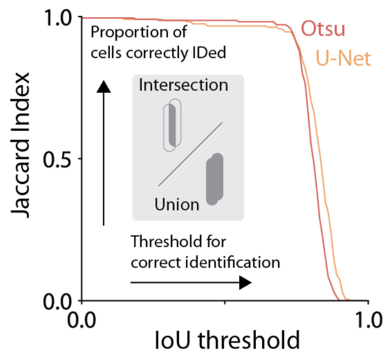
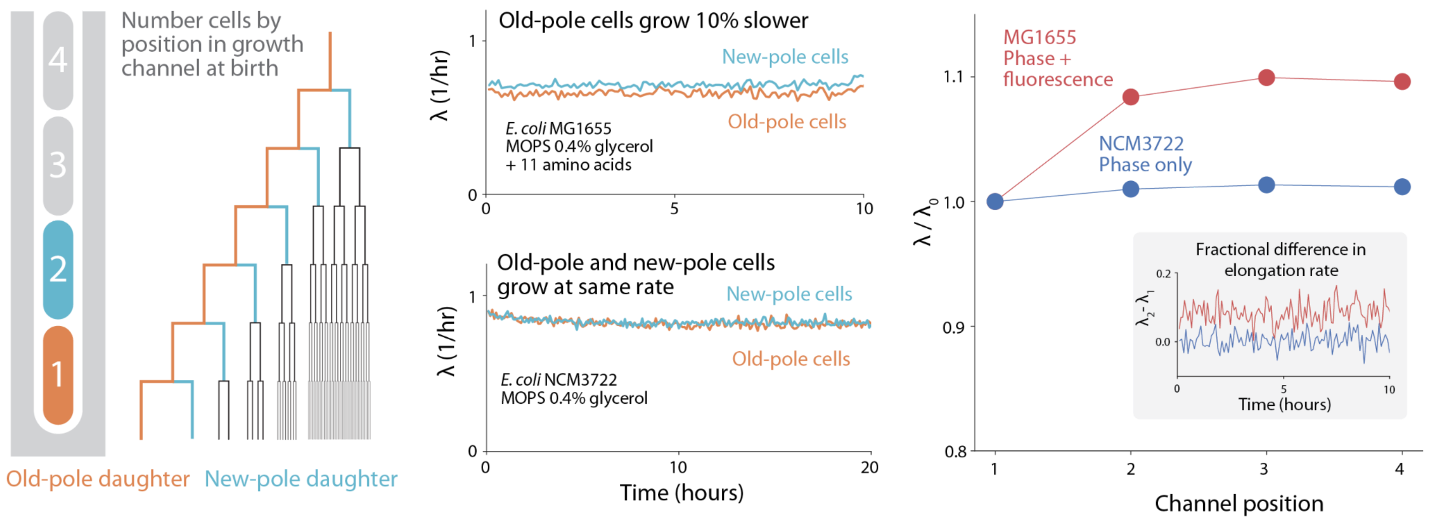


Figure S3: Old-pole aging phenotype is strain specific. Cells imaged with fluorescence often show signs of aging in the old-pole "mother" cell. For instance, in the dataset analyzed in Figure 4 (*E. coli* MG1655 with the fluorescent protein YPet fused to DnaN), we observed systematic differences in cell elongation rate and size between the old-pole cell at the end of the growth channel and its sisters, which inherit the new pole (top center). However, this asymmetry is not universal. Using napari-MM3's Otsu segmentation method, we re-analyzed previously published data obtained without fluorescence illumination [30], and found that the old-pole and new-pole cell elongation rates varied only on the order of 1% (lower center), while in the dataset obtained under fluorescence imaging, the old-pole mother cells grow 7-10% slower than the new pole cells.



References

1. Wang P, Robert L, Pelletier J, Dang WL, Taddei F, Wright A, et al. Robust growth of Escherichia coli. Curr Biol. 2010;20: 1099–1103.
2. Sauls JT, Cox SE, Do Q, Castillo V, Ghulam-Jelani Z, Jun S. Control of Bacillus subtilis Replication Initiation during Physiological Transitions and Perturbations. MBio. 2019;10. doi:10.1128/mBio.02205-19
3. Crane MM, Clark IBN, Bakker E, Smith S, Swain PS. A microfluidic system for studying ageing and dynamic single-cell responses in budding yeast. PLoS One. 2014;9: 1–10.
4. Nakaoka H, Wakamoto Y. Aging, mortality, and the fast growth trade-off of Schizosaccharomyces pombe. PLoS Biol. 2017;15: 1–29.
5. Jun S, Si F, Pugatch R, Scott M. Fundamental principles in bacterial physiology—history, recent progress, and the future with focus on cell size control: a review. Rep Prog Phys. 2018;81: 056601.
6. Bakshi S, Leoncini E, Baker C, Cañas-Duarte SJ, Okumus B, Paulsson J. Tracking bacterial lineages in complex and dynamic environments with applications for growth control and persistence. Nat Microbiol. 2021;6: 783–791.
7. Kaplan Y, Reich S, Oster E, Maoz S, Levin-Reisman I, Ronin I, et al. Observation of universal ageing dynamics in antibiotic persistence. Nature. 2021;600: 290–294.
8. Russell JR, Cabeen MT, Wiggins PA, Paulsson J, Losick R. Noise in a phosphorelay drives stochastic entry into sporulation in Bacillus subtilis. EMBO J. 2017;36: 2856–2869.
9. Amir A, Babaeipour F, McIntosh DB, Nelson DR, Jun S. Bending forces plastically deform growing bacterial cell walls. Proc Natl Acad Sci U S A. 2014;111: 5778–5783.
10. Stewart EJ, Madden R, Paul G, Taddei F. Aging and death in an organism that reproduces by morphologically symmetric division. PLoS Biol. 2005;3: e45.
11. O'Connor OM, Alnahhas RN, Lugagne J-B, Dunlop MJ. DeLTA 2.0: A deep learning pipeline for quantifying single-cell spatial and temporal dynamics. PLoS Comput Biol. 2022;18: e1009797.
12. Sachs CC, Grünberger A, Helfrich S, Probst C, Wiechert W, Kohlheyer D, et al. Image-Based Single Cell Profiling: High-Throughput Processing of Mother Machine Experiments. PLoS One. 2016;11: e0163453.
13. Smith A, Metz J, Pagliara S. MMHelper: An automated framework for the analysis of microscopy images acquired with the mother machine. Sci Rep. 2019;9: 10123.
14. Ollion J, Elez M, Robert L. High-throughput detection and tracking of cells and intracellular spots in mother machine experiments. Nat Protoc. 2019;14: 3144–3161.
15. Stylianidou S, Brennan C, Nissen SB, Kuwada NJ, Wiggins PA. SuperSegger: robust image segmentation, analysis and lineage tracking of bacterial cells. Mol Microbiol. 2016;102: 690–700.
16. Panigrahi S, Murat D, Le Gall A, Martineau E, Goldlust K, Fiche J-B, et al. Misic, a general deep learning-based method for the high-throughput cell segmentation of complex bacterial communities. Elife. 2021;10. doi:10.7554/eLife.65151
17. Cutler KJ, Stringer C, Wiggins PA, Mougous JD. Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation. Nat Methods. 2022; 2021.11.03.467199.
18. Spahn C, Gómez-de-Mariscal E, Laine RF, Pereira PM, von Chamier L, Conduit M, et al. DeepBacs for multi-task bacterial image analysis using open-source deep learning approaches. Commun Biol. 2022;5: 688.

19. Moen E, Borba E, Miller G, Schwartz M, Bannon D, Koe N, et al. Accurate cell tracking and lineage construction in live-cell imaging experiments with deep learning. bioRxiv. 2019. p. 803205. doi:10.1101/803205

20. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. Lect Notes Comput Sci. 2015;9351: 234–241.

21. Si F, Le Treut G, Sauls JT, Vadia S, Levin PA, Jun S. Mechanistic Origin of Cell-Size Control and Homeostasis in Bacteria. Curr Biol. 2019;29: 1760–1770.e7.

22. Taheri-Araghi S, Brown SD, Sauls JT, McIntosh DB, Jun S. Single-Cell Physiology. Annu Rev Biophys. 2015;44: 123–142.

23. Allard P, Papazotos F, Potvin-Trottier L. Microfluidics for long-term single-cell time-lapse microscopy: Advances and applications. Front Bioeng Biotechnol. 2022;10: 968342.

24. Potvin-Trottier L, Luro S, Paulsson J. Microfluidics and single-cell microscopy to study stochastic processes in bacteria. Curr Opin Microbiol. 2018;43: 186–192.

25. Cabeen MT, Losick R. Single-cell Microfluidic Analysis of Bacillus subtilis. J Vis Exp. 2018. doi:10.3791/56901

26. Kamande JW, Wang Y, Taylor AM. Cloning SU8 silicon masters using epoxy resins to increase feature replicability and production for cell culture devices. Biomicrofluidics. 2015;9: 036502.

27. mother-machine-protocols: Procedures for duplicating, constructing and using the microfluidic mother machine device. Github; Available: https://github.com/junlabucsd/mother-machine-protocols

28. Napari hub. [cited 17 Jan 2023]. Available: https://www.napari-hub.org/plugins/napari-mm3

29. napari — napari. [cited 23 Jan 2023]. Available: https://napari.org/stable/

30. Taheri-Araghi S, Bradde S, Sauls JT, Hill NS, Levin PA, Paulsson J, et al. Cell-size control and homeostasis in bacteria. Curr Biol. 2015;25: 385–391.

31. Sauls JT, Schroeder JW, Si F, Brown SD, Le Treut G, Wang JD. Mother machine image analysis with MM3. bioRxiv. 2019; 4–7.

32. Otsu N. A Threshold Selection Method from Gray-Level Histograms. IEEE Trans Syst Man Cybern. 1979;9: 62–66.

33. Grady L. Random walks for image segmentation. IEEE Trans Pattern Anal Mach Intell. 2006;28: 1768–1783.

34. notebooks at main · junlabucsd/napari-mm3. Github; Available: https://github.com/junlabucsd/napari-mm3

35. Banerjee DS, Stephenson G, Das SG. Segmentation and analysis of mother machine data: SAM. bioRxiv. 2020. p. 2020.10.01.322685. doi:10.1101/2020.10.01.322685

36. Lugagne JB, Lin H, Dunlop MJ. Delta: Automated cell segmentation, tracking, and lineage reconstruction using deep learning. PLoS Comput Biol. 2020;16: 1–18.

37. Kaiser M, Jug F, Julou T, Deshpande S, Pfohl T, Silander OK, et al. Monitoring single-cell gene regulation under dynamically controllable conditions with integrated microfluidics and software. Nat Commun. 2018;9. doi:10.1038/s41467-017-02505-0

38. Stringer C, Wang T, Michaelos M, Pachitariu M. Cellpose: a generalist algorithm for cellular segmentation. Nat Methods. 2021;18: 100–106.

39. Ollion J, Ollion C. DistNet: Deep Tracking by Displacement Regression: Application to Bacteria Growing in the Mother Machine. Medical Image Computing and Computer Assisted Intervention – MICCAI 2020. Springer International Publishing; 2020. pp. 215–225.

40. mother-machine-data: A repository for processed mother machine data from the Jun Lab. Github; Available: https://github.com/junlabucsd/mother-machine-data

41. Ollion J. bacmman. Github; Available: https://github.com/jeanollion/bacmman

42. Le Treut G, Si F, Li D, Jun S. Quantitative Examination of Five Stochastic Cell-Cycle and Cell-Size Control Models for Escherichia coli and Bacillus subtilis. Front Microbiol. 2021;12: 721899.

43. Geiger RS, Cope D, Ip J, Lotosh M, Shah A, Weng J, et al. "Garbage in, garbage out" revisited: What do machine learning application papers report about human-labeled training data? Quant Sci Stud. 2021;2: 795–827.

44. Laine RF, Arganda-Carreras I, Henriques R, Jacquemet G. Avoiding a replication crisis in deep-learning-based bioimage analysis. Nat Methods. 2021;18: 1136–1144.

45. Image.Sc forum. In: Image.sc Forum [Internet]. [cited 24 Jan 2023]. Available: https://forum.image.sc/

46. Bourne R. ImageJ. In: Bourne R, editor. Fundamentals of Digital Imaging in Medicine. London: Springer London; 2010. pp. 185–188.

47. Assets — DeLTA 2.0-gamma documentation. [cited 23 Feb 2023]. Available: https://delta.readthedocs.io/en/latest/usage/assets_desc.html

48. Shiaelis N, Tometzki A, Peto L, McMahon A, Hepp C, Bickerton E, et al. Virus detection and identification in minutes using single-particle imaging and deep learning. ACS Nano. 2023;17: 697–710.

49. mother-machine-data: A repository for processed mother machine data from the Jun Lab. Github; Available: https://github.com/junlabucsd/mother-machine-data

50. Seita A, Nakaoka H, Okura R, Wakamoto Y. Intrinsic growth heterogeneity of mouse leukemia cells underlies differential susceptibility to a growth-inhibiting anticancer drug. PLoS One. 2021;16: e0236534.

51. Pearl Mizrahi S, Gefen O, Simon I, Balaban NQ. Persistence to anti-cancer treatments in the stationary to proliferating transition. Cell Cycle. 2016;15: 3442–3453.

52. Lawson MJ, Camsund D, Larsson J, Baltekin Ö, Fange D, Elf J. In situ genotyping of a pooled strain library after characterizing complex phenotypes. Mol Syst Biol. 2017;13: 947.

53. Luro S, Potvin-Trottier L, Okumus B, Paulsson J. Isolating live cells after high-throughput, long-term, time-lapse microscopy. Nature Methods. 2020. pp. 93–100. doi:10.1038/s41592-019-0620-7

54. Videos JL. Mother Machine Data Analysis with Napari. Youtube; 6 Oct 2022 [cited 25 Mar 2023]. Available: https://www.youtube.com/watch?v=7MCiGTg6mq4

55. mother-machine-data: A repository for processed mother machine data from the Jun Lab. Github; Available: https://github.com/junlabucsd/mother-machine-data

56. mother-machine-protocols: Procedures for duplicating, constructing and using the microfluidic mother machine device. Github; Available: https://github.com/junlabucsd/mother-machine-protocols

57. mother-machine-data: A repository for processed mother machine data from the Jun Lab. Github; Available: https://github.com/junlabucsd/mother-machine-data

58. Lee TC, Kashyap RL, Chu CN. Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms. CVGIP: Graphical Models and Image Processing. 1994;56: 462–478.

59. Jeckel H, Drescher K. Advances and opportunities in image analysis of bacterial cells and communities. FEMS Microbiol Rev. 2021;45. doi:10.1093/femsre/fuaa062