

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

2 Ryan Thiermann^{1*}, Michael Sandler^{1*}, Gursharan Ahir^{1*}, John T. Sauls^{1*}, Jeremy W. Schroeder^{2*}, Steven D. Brown¹,
3 Guillaume Le Treut³, Fangwei Si⁴, Dongyang Li⁵, Jue D. Wang⁶, Suckjoon Jun^{1**}

4

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

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

7 ³ Chan Zuckerberg Biohub, San Francisco, CA

8 ⁴ Department of Physics, Carnegie Mellon University, Pittsburgh, PA

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

10 ⁶ Department of Bacteriology, University of Wisconsin-Madison, Madison, WI

11

12 *These authors contributed equally to this work

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

14

15 Abstract

16 Despite much progress, image processing remains a significant bottleneck for high-throughput analysis of microscopy
17 data. One popular platform for single-cell time-lapse imaging is the mother machine, which enables long-term tracking of
18 microbial cells under precisely controlled growth conditions. While several mother machine image analysis pipelines have
19 been developed in the past several years, adoption by a non-expert audience remains a challenge. To fill this gap, we
20 implemented our own software, MM3, as a plugin for the multidimensional image viewer napari. napari-MM3 is a
21 complete and modular image analysis pipeline for mother machine data, which takes advantage of the high-level
22 interactivity of napari. Here, we give an overview of napari-MM3 and test it against several well-designed and
23 widely-used image analysis pipelines, including BACMMAN and DeLTA. Researchers often analyze mother machine data
24 with custom scripts using varied image analysis methods, but a quantitative comparison of the output of different
25 pipelines has been lacking. To this end, we show that key single-cell physiological parameter correlations and
26 distributions are robust to the choice of analysis method. However, we also find that small changes in thresholding
27 parameters can systematically alter parameters extracted from single-cell imaging experiments. Moreover, we explicitly
28 show that in deep learning based segmentation, “what you put is what you get” (WYPIWYG) - i.e., pixel-level variation in
29 training data for cell segmentation can propagate to the model output and bias spatial and temporal measurements.
30 Finally, while the primary purpose of this work is to introduce the image analysis software that we have developed over
31 the last decade in our lab, we also provide information for those who want to implement mother-machine-based
32 high-throughput imaging and analysis methods in their research.

33 Introduction

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

42 Despite the progress in imaging techniques and microfluidics, image processing remains a major bottleneck in the
 43 analysis pipelines. The unique structure of the mother machine device enables precise control of growth conditions and
 44 long-term tracking of cells, to the degree that cannot be achieved by traditional tracking of cells in microcolonies [11].
 45 However, automated image processing is essential to process the large amounts of data generated by these
 46 high-throughput experiments. In addition, the unique structure of the mother machine device requires a specialized
 47 workflow to select and track individual growth channels. As experimentalists often need to extract precise statistics over
 48 multiple generations or observe rare events, the analysis workflow must be modular to allow inspection and curation of
 49 intermediate results. To meet these needs, numerous mother machine-specific image analysis packages have been
 50 introduced in the last few years [12–15], in addition to general image analysis packages adaptable to the mother machine
 51 workflow [16–20]. Much recent work has been catalyzed by advances in biomedical image analysis with deep
 52 convolutional neural networks, particularly the U-Net architecture [21]. Many of these tools [15,22] have been designed
 53 with ease-of-use and accessibility in mind. However, they can still present a steep learning curve for first-time users. In
 54 addition, as the outputs of these pipelines are often used by researchers to derive biological principles based on
 55 correlations, it is important to understand the limitations of and differences between different image analysis methods.

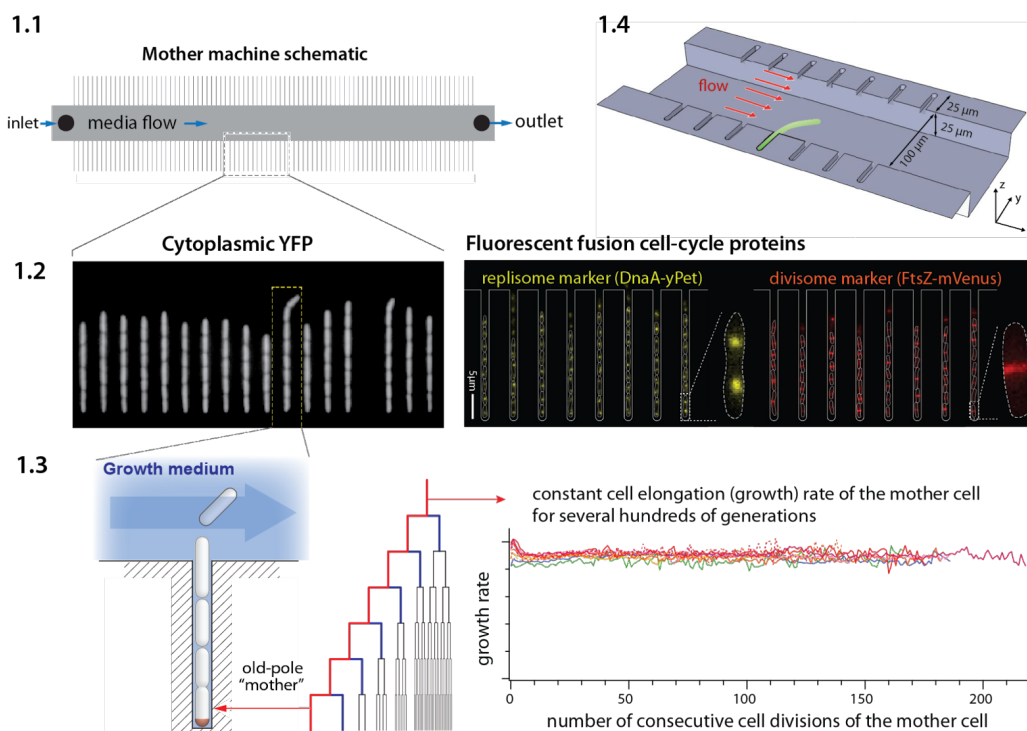


Figure 1: Mother machine workflow, schematic, and applications. (1.1) Mother machine schematic. Growth channels flank a central flow cell that supplies fresh media and whisks away daughter cells. In a typical experiment, numerous fields of view (FOVs) are imaged for several hours. (1.2) Fluorescence images of *E. coli* strains expressing cytoplasmic YFP [1] (left) and markers for the replisome protein DnaN and division protein FtsZ (right) [10]. (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].

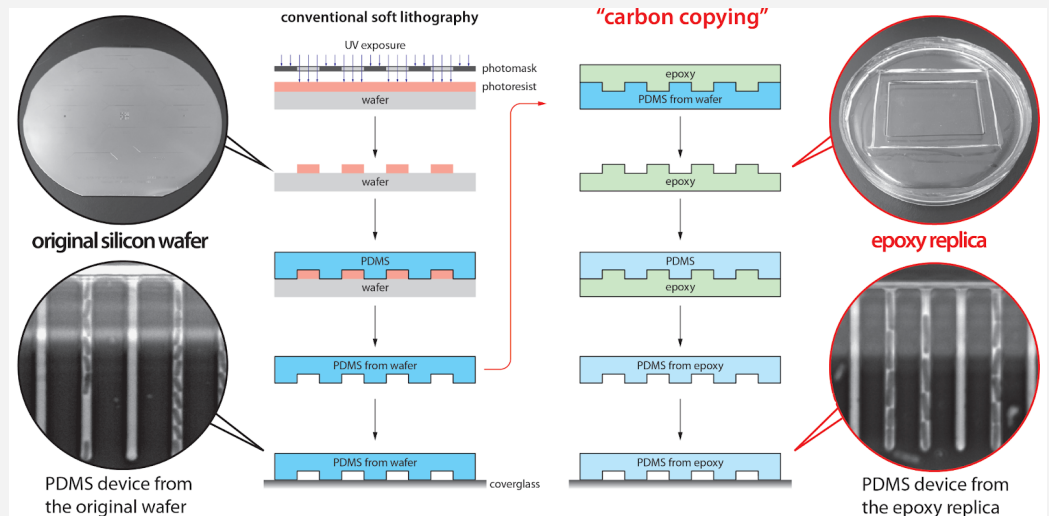
Box 1: Mother machine experimental workflow

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 [23], along with other recent reviews [24,25] and published protocols [26], 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 [27], 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 [28].



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 [28] (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.

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

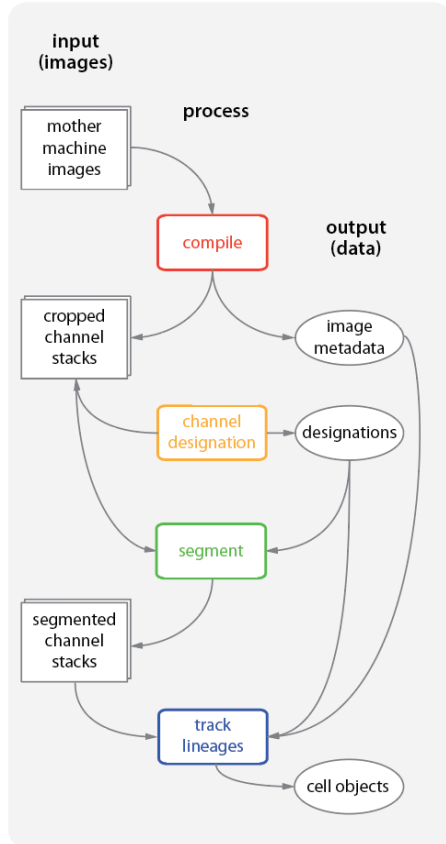
106

107

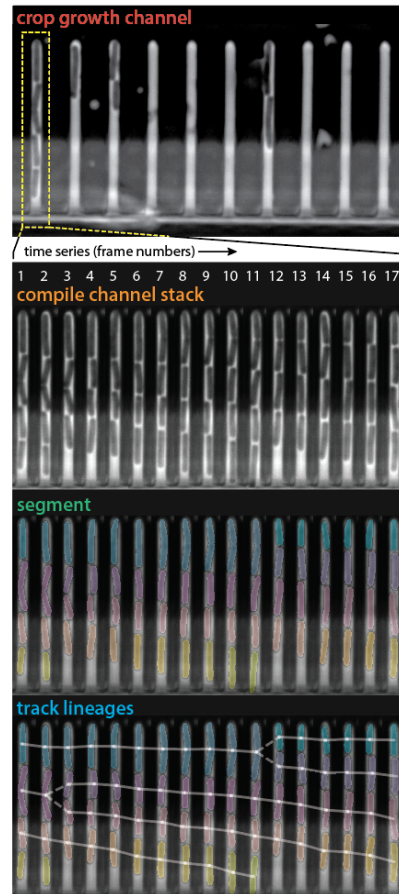
108

2.1

MM3 workflow



2.2 Example image analysis process



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 [29], 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 [30]. 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. With this in mind, we show that “what you put is what you get”: both classical and deep learning-based segmentation methods are highly sensitive to user-determined threshold values. As exact cell boundaries may be difficult to distinguish by eye, these values are difficult to set definitively, and can systematically alter the output of the analysis. Fortunately, we find that key single-cell physiological parameter correlations and distributions are robust to the choice of analysis method. However, interpreting and comparing the results of different analyses requires care.

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. 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 comprising hundreds of gigabytes of data. While the experimental methods for mother machine experiments have become increasingly accessible, image analysis tools have lagged behind. 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.

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

109 flexible solution for mother machine image analysis, taking raw images and producing readily graphable cell data, while
110 accommodating both machine learning-based and traditional computer vision techniques. It supports phase contrast
111 and fluorescence images, and has been tested with different species (bacteria *E. coli* and *B. subtilis*, yeast *S. pombe*),
112 mother machine designs, and optical configurations. The modular pipeline architecture allows flexible use of mid-stream
113 outputs and straightforward troubleshooting (for instance, while *M. mycoides* is too small to segment with traditional
114 microscopy methods [31], we were able to obtain growth rate measurements by running the first half of the pipeline).

115 MM3 reflects the culmination of several iterations of our in-house mother machine analysis software developed over the
116 past decade. Before MM3, we developed our image analysis pipeline in C++ [1] and MATLAB [32]. Eventually, Python
117 became enormously popular, and we began MM3 as a set of Python scripts run from the command line [33]. However,
118 the command-line-based interface had several drawbacks. The interface was more difficult for users unfamiliar with the
119 command line or programming. It also had limited interactivity. As a result, troubleshooting was difficult and required
120 modifying the source code to display image output at intermediate steps or manually inspecting output files in ImageJ.
121 This made the user repeatedly move back and forth between different windows and applications, slowing the analysis.

122 These drawbacks motivated us to convert MM3 into a plug-in for the Python-based interactive image viewer napari [30].
123 napari provides an N-dimensional display ideal for visualizing multichannel time-lapse data. It offers built-in annotation
124 tools and label layers to compare and annotate segmentation masks and tracking labels. It also provides a Python
125 interpreter, allowing users to move easily between the viewer interface and the underlying data objects. For the best
126 usability, we designed the napari-MM3 plug-in to allow the user to run the entire pipeline without leaving the napari
127 interface.

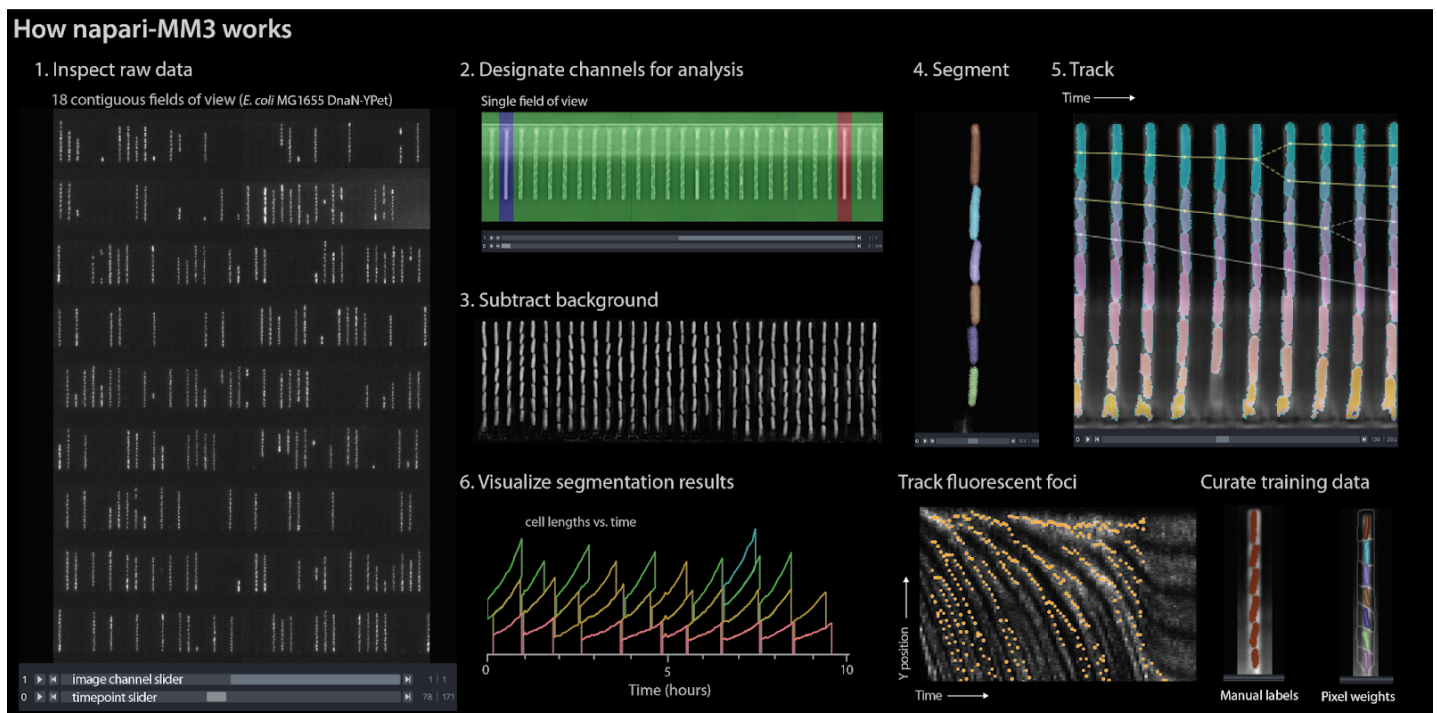


Figure 3: napari-MM3 interface. The napari viewer enables interactive analysis of mother machine data with real-time feedback and fast debugging. Raw data shown is from MG1655 background *E. coli* expressing the fluorescence protein YPet fused to the replisome protein DnaN [10].

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

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

134 We elaborate on these steps as follows.

135 1. Channel detection and curation

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

142

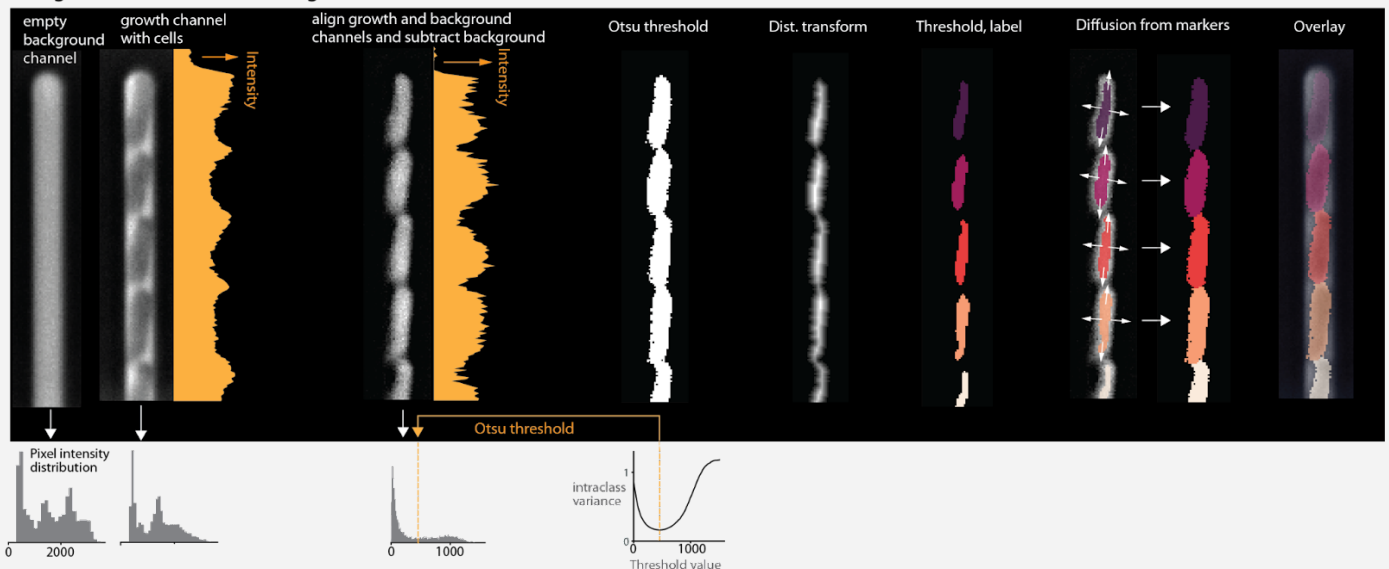
143 2. Cell segmentation

144 napari-MM3 offers two methods for cell segmentation, one using traditional computer vision techniques and the other
145 using deep learning. The non-learning method utilizes Otsu's method to apply a binary threshold to separate cell objects
146 from the background. It then labels the isolated cells and uses a random walker algorithm [34] to fill out the cell
147 boundaries. This method is fast but optimized for specific mother machine designs and phase contrast imaging of
148 bacteria. It also requires accurate background subtraction of phase contrast images (Box 2), to ensure that the presence
149 of the channel border does not interfere with cell detection. The supervised learning method uses a convolutional neural
150 net (CNN) with the U-Net architecture [21,22,35]. The napari viewer can be used to construct training data, with the
151 option to import existing Otsu or U-Net segmentation output as a template. The neural net can then be trained directly
152 from napari, with the option to check the performance of the model in the napari viewer after successive rounds of
153 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 that 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 [36] 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 [34] on the original image.

Background subtraction and segmentation via Otsu's method



154

155 3. Cell tracking and lineage reconstruction

156 Finally, napari-MM3 links segmented cells in time to define a lineage of cell objects, using a simple decision tree based
157 on a priori knowledge of binary fission and the mother machine. Tracking produces a dictionary of cell objects containing
158 relevant information derived from the cell segments, including the cell lengths and volumes over time, cell elongation
159 rate, and generation time. Plotting and additional analysis can then be done with the user's tool of choice. Statistics can

160 be directly extracted from the cell objects, or the cell objects can be converted into a .csv file, a pandas DataFrame, or a
161 MATLAB structure. We provide a Jupyter notebook demonstrating this analysis at [37].

162 Additional features and future extensions

163 napari-MM3 offers several additional modules supplemental to the main processing pipeline, including methods for
164 fluorescence image analysis and U-Net training data construction and model training. Integrated fluorescence signal and
165 fluorescence per cell area and volume for each timepoint can be extracted using the “Colors” module. napari-MM3 also
166 includes a module for the detection and tracking of fluorescent spots or “foci.” For example, we have used it to track
167 fluorescently labeled replisome machinery in bacteria in order to measure the timing and synchrony of DNA replication
168 initiation [2,10]. Lastly, U-Net segmentation training data can be constructed by manual annotation of raw images in the
169 napari viewer. napari-MM3 offers the option to construct training data with existing Otsu or U-Net segmentation data as
170 a template. This allows the user to iteratively train a model, correct mistakes in its output, and use the modified output as
171 input for the next round of training. We also provide a Jupyter notebook covering training data construction and model
172 training at [37].

173 Going forward, we plan to add support for additional segmentation and tracking modalities [18,38]. We will also
174 incorporate support for additional organisms such as the budding yeast *S. cerevisiae*. Finally, we plan to take advantage
175 of napari’s interactive display to add interactive data visualization and plotting.

176 Performance test of napari-MM3

177 To evaluate the speed of napari-MM3, we timed the processing of a typical dataset (Table 1). Using consumer-grade
178 hardware, a single-channel stack consisting of several hundred time frames can be processed in less than five seconds,
179 and a typical experiment consisting of 25 GB of imaging data can be processed in under an hour. These metrics are on
180 par with those reported by other recently published mother machine software [15,22,40].

	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

181 **Table 1: Performance metrics for napari-MM3.** Processing times were measured on an iMac with a 3.6 GHz 10-Core Intel Core i9
182 processor with 64 GB of RAM and an AMD Radeon Pro 5500 XT 8 GB GPU. Tensorflow was configured to use the AMD GPU
183 according to [39]. The GPU was used in U-Net training and segmentation steps. The dataset analyzed is from [10] and consists of 26
184 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
185 Otsu segmentation method is slightly faster than the U-Net, it also requires a background subtraction step, such that the total
186 runtimes of the two methods are comparable.

187 Testing napari-MM3 on other published datasets

188 We tested napari-MM3 on several publicly available mother machine datasets: three from experiments with *E. coli*
189 [15,22,41] and one from *C. glutamicum* [13]. We were able to process all 4 datasets with minimal adjustments to the
190 default parameter values (Methods). We quantified the performance of MM3 on each dataset by comparing the output of
191 the Otsu segmentation method to manually determined ground truth masks from a subset of each dataset (Table 2). We
192 computed the segmentation accuracy as the Jaccard Index at an IoU threshold of 0.6 (Methods). The software
193 performed well on BACMMAN, molyso and MoMa datasets, with segmentation accuracies of 97.9%, 98.4% and 100.0%
194 respectively. Segmentation was notably worse on the DeLTA dataset, with an accuracy of 91.5%. However, we observed
195 that the majority of segmentation errors arose from misclassification of cells near the channel opening, where
196 determining cell boundaries is often more difficult.

197 Comparison with other image analysis software

198 We also tested napari-MM3’s usability and performance against other popular software. We began by surveying a range
199 of existing mother machine image analysis tools (Table 3). Some early analysis pipelines used one-dimensional

200 segmentation methods [13,41], which perform adequately when cells are tightly confined in the growth channels. In
201 recent years, many excellent general-purpose CNN-based cell segmentation tools have also been developed [16–19,42],
202 which may be extended to process mother machine data.

203

Dataset	Correctly segmented cells	False positives	False negatives	Accuracy
BACMMAN [15]	228	4	1	97.9%
DeLTA [12,22]	247	22	1	91.5%
molyso [13]	247	4	0	98.4%
MoMa [41]	80	0	0	100.0%

204 **Table 2: Testing napari-MM3 on external datasets.** Segmentation accuracy of napari-MM3 on published datasets from other
205 groups [13,15,22,41]. As exact boundaries are difficult to determine by eye, a cell was considered to be correctly segmented if
206 the Intersection over Union of the predicted mask and ground truth mask was greater than 0.6 (Methods). The accuracy is the
207 Jaccard Index i.e. the ratio of true positives (correctly identified cells) to the sum of true positives, false positives (identified cells
208 which were not present in the ground truth data) and false negatives (ground truth cells which were not identified by the
209 segmentation).

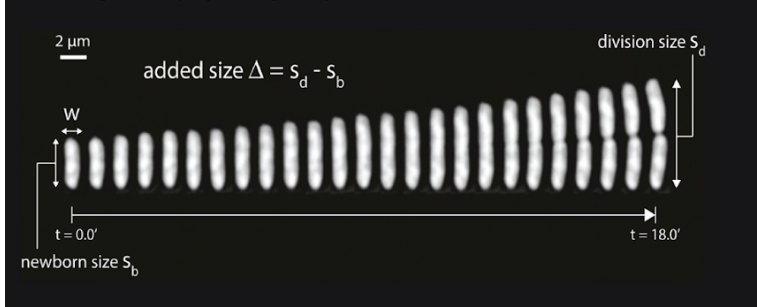
210 In this work, we only tested mother-machine-specific pipelines. In particular, we constrained our analysis to DeLTA and
211 BACMMAN, two excellent open-source mother machine-specific pipelines offering 2D segmentation and cell tracking,
212 which are also well-documented and actively maintained. BACMMAN [15] performs 2D segmentation via traditional
213 computer vision methods similar to those implemented in napari-MM3 and has recently added support for CNN-based
214 segmentation as well [43]. DeLTA [12,22] uses the U-Net architecture for channel detection, cell segmentation, and cell
215 tracking, with a mother machine-specific and general agar pad mode. We used BACMMAN, DeLTA, and napari-MM3 to
216 analyze the same published dataset [10] consisting of *E. coli* MG1655 grown in minimal growth medium (MOPS 0.4%
217 glycerol + 11 amino acids with ~60 minute doubling time) [10]. Data processed in napari-MM3 was separately
218 segmented with U-Net and traditional computer vision methods. We found that the pre-trained mother machine model
219 provided with DeLTA did not generalize well to our data. However, after training a new model with representative data, we
220 achieved accurate segmentation.

Software	Implementation	Segmentation	Deep learning support
BACMMAN [15] / DistNet [38]	ImageJ plugin	2D	✓
DeLTA [12,22]	Python package	2D	✓
napari-MM3 [33], this work	napari plug-in	2D	✓
SAM [40]	MATLAB	2D	
MMHelper [14]	ImageJ plugin	2D	
molyso [13]	Python package	1D	
MoMa [41]	ImageJ plugin	1D	

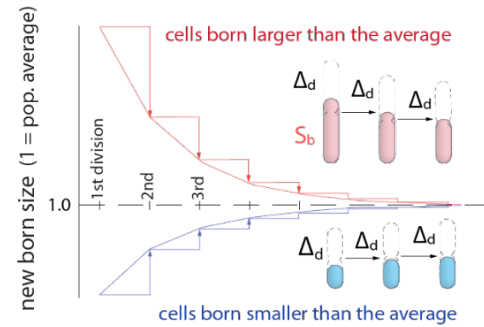
221 **Table 3: Overview of mother-machine image analysis tools.** A comparison of several published imaging methods. ‘2D’ or ‘1D’
222 segmentation indicates whether the cells are labeled in an image and analyzed in two dimensions, or projected onto a vertical
223 axis and analyzed in one dimension. Several tools support the use of deep learning (in place of or in addition to classical
224 computer vision techniques).

225

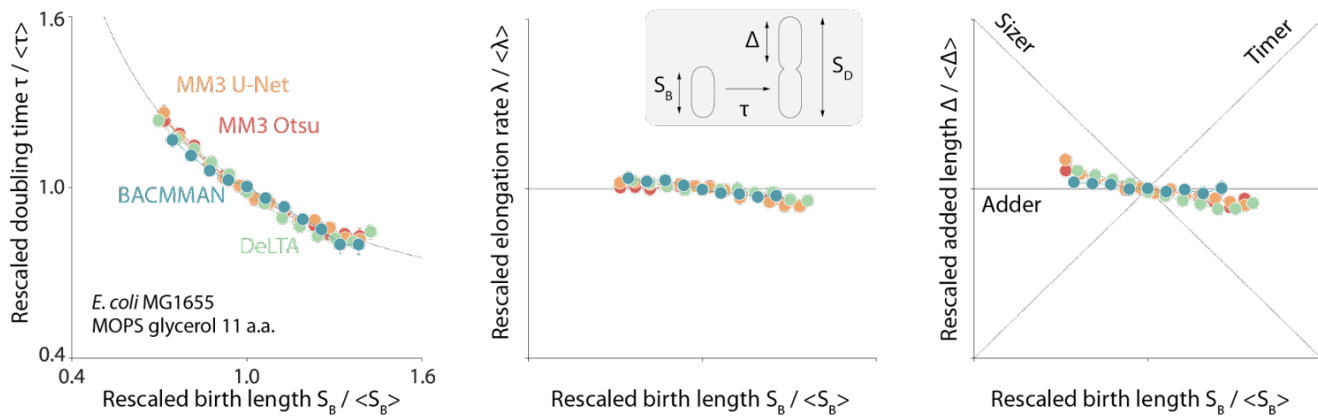
4.1 Single cell physiological parameter definitions



4.2 Cell size homeostasis via the adder principle



4.3 Adder correlations are robust to choice of analysis methods



4.4 Hierarchy of physiological parameter distributions is robust to choice of analysis method

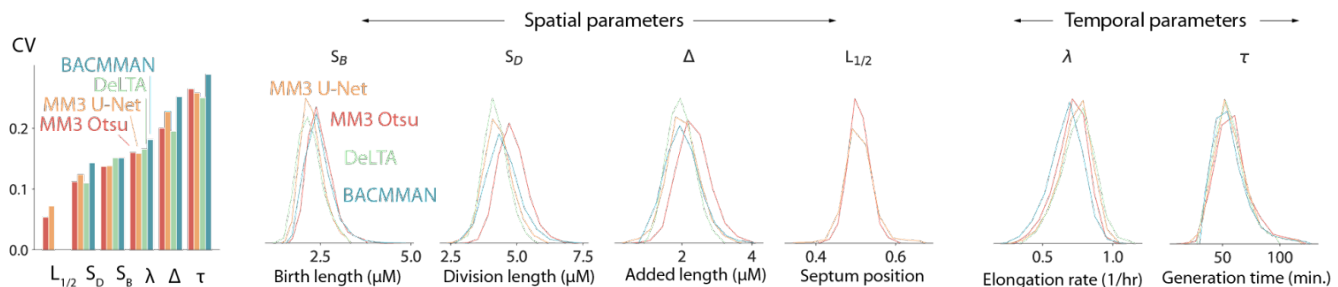


Figure 4: Comparison of various image analysis approaches. (4.1) A time series of a typical cell growing in a nutrient-rich medium. The birth size, division size, and added size are indicated. (4.2) The adder principle ensures cell size homeostasis via passive convergence of cell size to the population 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 [10] 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 [44]

226 We compared the distributions and correlations of key physiological parameters generated by each analysis tool,
 227 motivated by our standard approach to single-cell physiology [5,10,32,45]. First, we confirmed that all four analysis
 228 methods yield essentially identical correlations between cell length at birth (S_B) vs. (a) generation time (τ), (b) elongation
 229 rate (λ), and (c) the length added between birth and division (Δ) (Figure 4.3). Next, we compared the distributions of
 230 various physiological parameters. The CV (coefficient of variation) of a physiological parameter distribution is often taken
 231 to reflect the tightness of the underlying biological control. We have previously found [2,32] that the CVs of a set of
 232 physiological parameters (birth length, division length, length added between divisions, growth rate, generation time, and
 233 septum position) are invariant across growth conditions in *E. coli* and *B. subtilis*, and that the hierarchy of CVs is
 234 preserved across the two evolutionarily divergent species [2,32]. Here, we confirmed that the distributions of these

235 physiological parameters are independent of the analysis methods (Figure 4.4). In particular, the hierarchy of CVs is
236 preserved by all three methods tested. Last, while in this dataset the old-pole “mother” cells showed signs of aging (in
237 particular, a reduced elongation rate), this aging phenotype is strain- and condition-dependent (Figure S3).

238 **Systematic discrepancies in cell segmentation outputs**

239 While we found that the correlations between physiological parameters were preserved across the different analysis
240 methods (Figure 4.3), we also observed systematic discrepancies in the results obtained by different methods, including
241 cell length at birth (S_b), length at division (S_d), and length added between birth and division (Δ) (Figure 4.4). In particular,
242 napari-MM3’s classical segmentation method systematically generated larger cell masks than napari-MM3 U-Net,
243 DeLTA, and BACMMAN (Figure 4.4). We focused on the discrepancies between the two MM3 outputs. Although the
244 deviation between the two masks may not appear significant when individual masks are inspected by eye (Figure 5.1,
245 Figure S2), the classical method yields cells that are 5%-10% larger at each time point than those returned by the U-Net
246 method when averaging over an entire experiment with tens of thousands of cells tracked (Figure 5.2). Cell birth and
247 division times are also systematically shifted in the classical method, as the expanded cell boundaries lead the algorithm
248 to split cells 1-2 time frames later on average.

249 The root of this discrepancy is as follows. Exact cell boundaries are difficult to distinguish by eye, and the classical
250 methods tested here require the user to set threshold values that can systematically alter the measured cell size. Indeed,
251 both MM3 and BACMMAN’s non-learning method (which also uses Otsu thresholding and a watershed / diffusion
252 algorithm) - output different cell masks with their ‘default’ parameter settings. On the other hand, binary U-Net
253 segmentation methods, such as those implemented in napari-MM3 and DeLTA, tend to output smaller cell sizes because
254 the model must leave a gap between cells so that they are not stitched together (note this is not a fundamental limitation
255 of U-Net, but a consequence of our implementation: see, e.g. [17] or [38] for more complex approaches which avoid this
256 issue).

257 **WYPIWYG (“What You Put Is What You Get”) in deep-learning-based image analysis**

258 Given that classical methods are clearly sensitive to this threshold tuning, we predicted that deep-learning approaches
259 would also be impacted [46,47]. We chose the recent cutting-edge segmentation model Omnipose and separately
260 trained it on masks derived from the aforementioned Otsu segmentation output and masks from the napari-MM3 U-Net
261 segmentation output. We chose Omnipose as it assigns different labels to different cells, and can thus segment cells
262 with contiguous boundaries, in contrast to MM3 or DeLTA’s U-Net implementations. Indeed, we found that the
263 systematic discrepancy in the training masks propagated to the output of the trained models: the Omnipose model
264 trained on larger Otsu masks generated larger masks upon evaluation with the same data, while the Omnipose model
265 trained on smaller U-Net masks output smaller masks (Figure 5.3). In computer science, the phrase “Garbage in,
266 garbage out” denotes the concept that undesirable attributes in the input to a program will propagate to the output
267 [48,49]. Here we propose a related notion WYPIWYG, or “what you put is what you get”. That is, at least for our setup,
268 systematic differences in training data masks lead the model to learn different threshold intensity values and thus to
269 systematically output larger or smaller masks. We emphasize this result does not reflect a flaw in Omnipose - whose
270 performance we found impressive - but rather a well-studied feature of machine learning methods in general [46].

271 **Discussion**

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

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

278 **Validating results**

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

282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333

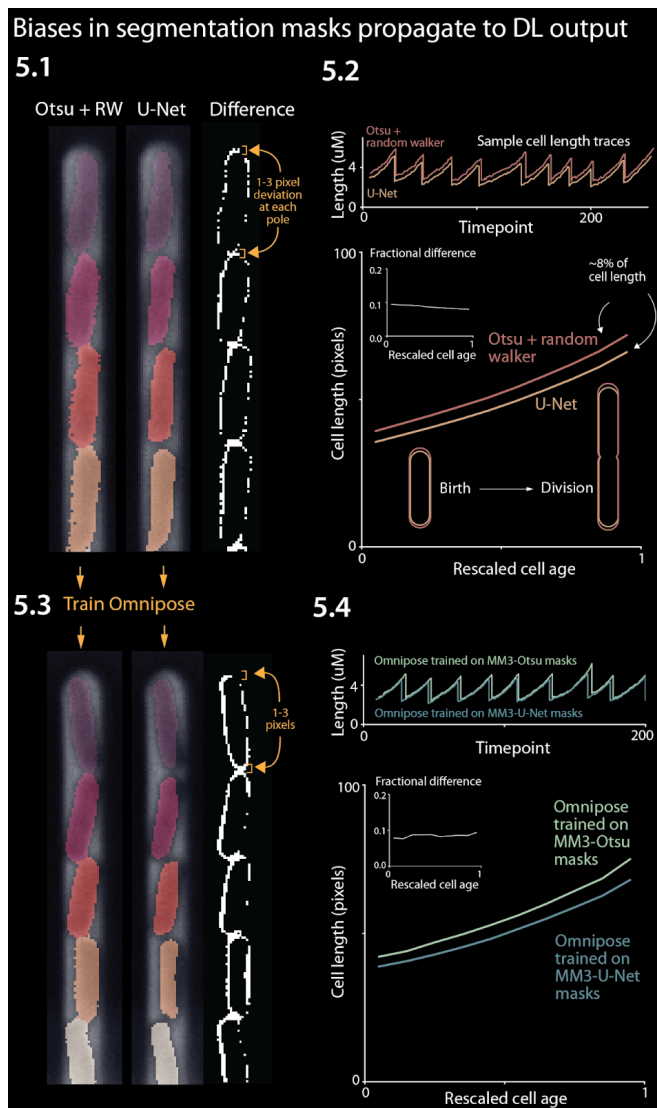


Figure 5: Effect of systematic deviation in segmentation output from different methods. 5.1. Otsu / random walker and U-Net segmentation masks. The classical method systematically yields masks that are 5%-10% larger than the other methods. 5.2. We confirmed that this discrepancy occurs consistently across 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.

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 [51] 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 [52] 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 [30] an attractive alternative to ImageJ.

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 [47] 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.

Inspired by previous reviews [47,50], we make the following suggestions for new users selecting a tool:

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

334 **Traditional computer vision vs. deep learning methods**

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

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

347 By contrast, the key strength of deep learning approaches is their ability to generalize to new conditions - whether to
348 different illumination conditions, different types of input images (phase contrast, brightfield, fluorescence) or different
349 organisms and cell types entirely. The main barrier to adoption of learning-based methods remains the construction of
350 training data, which can be tedious and time-consuming. A training data set of 50 - 100 images comprising several
351 hundred cells can be constructed in a few hours and will achieve passable segmentation on representative data.
352 However, larger training sets on the order of thousands of images are preferable, and will yield improved model accuracy
353 and generalizability. The time needed for annotation can be reduced by seeding the data with masks generated by
354 classical methods - or iteratively seeding with U-Net output - and then refining the masks further by hand. Model
355 performance and generalizability can often be significantly improved by augmenting training data via manipulations such
356 as rotating or shearing, distorting the intensity profile, and adding noise. Nonetheless, we have found that even with
357 extensive data augmentation, applying the U-Net segmentation to new experimental configurations or imaging
358 conditions often requires retraining the model on an expanded dataset with more representative data. Ultimately, deep
359 learning methods are only as good as the data they are trained on, and are most likely to fail when training data is
360 insufficient, mislabeled, or not representative. Going forward, sharing of training sets and models [53] between different
361 groups can facilitate progress and aid reproducibility.

362 In addition to deep learning-based segmentation, learning-based cell tracking in the mother machine has been
363 implemented recently by multiple groups [12,38]. For cells growing unconstrained on 2D surfaces such as agar pads,
364 U-Net tracking dramatically outperforms traditional methods [12]. However, for steady-state growth in the mother
365 machine where cells are confined and constrained to move in one dimension, we have not found a significant difference
366 between the performance of deep learning-based tracking and the non-learning tracking method implemented in MM3.
367 In both cases, errors in tracking nearly always arise from errors in segmentation. However, deep learning-based tracking
368 may offer an advantage in cases where cells may move substantially along the length of the channel, or undergo
369 dramatic morphological changes such as filamentation.

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

374 **Should users worry about the systematic discrepancy in segmentation results between different methods?**

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

378 If the research critically relies on the absolute cell size, such as cell-size control [10,32], the researcher must be aware of
379 inherent limitations to the accuracy of spatial measurements from cell segmentation. These arise in part from the
380 difficulty of consistently distinguishing cell boundaries by eye. Once a threshold is chosen, the choice will affect all
381 analyzed cells systematically. This limitation applies to both deep learning (through the construction of training data) and
382 traditional computer vision methods (through the manual input of a threshold value). For cell segmentation, the
383 uncertainties are typically comparable to the pixel size of the images, rather than optical resolutions. For example, the

384 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
385 non-negligible for many commonly cultured bacterial cells with submicron cell widths - e.g., *Enterobacteriales*,
386 *Pseudomonas*, *Bacillus subtilis*, and *Caulobacter crescentus*. For most commercially available cameras and objective
387 lenses used in quantitative bacterial cell biology, 10% should be taken as a conservative lower bound for uncertainty
388 when comparing absolute spatial measurements of bacterial cell size.

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

395 Fortunately, the examples mentioned above are extreme cases. For instance, the pixel-size uncertainties will reflect a
396 smaller proportion of the cell size when imaging larger cells such as yeast or mammalian cells. Even in our research on
397 single-cell bacterial physiology [2,10,32], we find that correlations and relative changes are more likely to be robust than
398 absolute spatial measurements to the choice of analysis method (Figure 4). Furthermore, different applications of
399 deep-learning based image analysis, such as high-throughput phenotypic classification [54] will be much more robust to
400 the pixel-size uncertainties in image segmentation results.

401 **Generating robust and unbiased segmentation results**

402 We have shown that both traditional computer vision and deep learning methods are susceptible to biases introduced by
403 imprecise thresholding and human error. How, then, can more precise cell boundaries be determined? For non-learning
404 methods, thresholds could be calibrated against data from alternate imaging methods such as fluorescence or
405 brightfield. For learning methods, one promising technique is the generation of synthetic training data [55]. This method
406 also has the advantage that new training datasets can be instantaneously for different imaging conditions or cell types,
407 once the appropriate parameters have been determined. For deep learning methods, metrics which lead the model to
408 recognize cell interiors or centers [18,38,56] may yield more robust results than binary pixel-level classification. Once cell
409 centers are known, boundaries can be determined relatively easily via classical watershed or random walker diffusion
410 algorithms.

411 **Conclusion and recommendations**

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

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

432 The mother machine setup has become increasingly accessible to researchers in recent years, through the distribution of
433 molds and the publication of in-depth protocols and open-source image analysis software. At the same time, new

434 variations of the device have found diverse applications, including bacterial starvation [6], and genetic screening [57,58].
435 Clearly, the combination of microfluidics with high-resolution time-lapse imaging remains powerful among single-cell
436 techniques. We hope that this article will prove useful to mother machine veterans and first-time users alike.

437 **Acknowledgements**

438 This work has been made possible in part by CZI grant DAF2021-239849 and grant DOI [59] from the Chan Zuckerberg
439 Initiative DAF, an advised fund of Silicon Valley Community Foundation (funder DOI 10.13039/100014989) . The work was
440 also supported by NIH grant R35GM139622 and NSF grant MCB-2016090.

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

443 Methods

444 Resources

- 445 • [napari-MM3 Github repository](#) [60].
- 446 ○ Contains installation instructions and video tutorial.
- 447 • [Jupyter notebook demonstrating analysis of MM3 output data](#). [37]
- 448 ○ A notebook providing functions for postprocessing and plotting of the napari-MM3 output
- 449 • [Protocols for device fabrication and loading](#) [28]
- 450 • [Raw and processed data analyzed in this manuscript](#) [44]

451

452 Getting started with napari-MM3

453 napari-MM3 is implemented entirely in Python and can be accessed on [Github](#) [60], along with documentation covering
454 installation and usage. It will run on a standard Mac, PC, or Linux machine. We recommend using the Anaconda Python
455 distribution to simplify installation.

456 Imaging conditions

457 The data analyzed in Figures 4 and 5 (originally published in [10]) was obtained on an inverted microscope (Nikon Ti-E)
458 with Perfect Focus 3 (PFS3), 100x oil immersion objective (PH3, numerical aperture = 1.45), and Obis laser 488LX
459 (Coherent Inc., CA) as a fluorescence light source, and an Andor NEO sCMOS (Andor Technology) camera. The laser
460 power was 18 mW. The exposure time was 200 ms for phase contrast imaging and 50 ms for fluorescence.

461 Image analysis for software comparison

462 For the software comparison in Figure 4, we analyzed a dataset from [10] consisting of *E. coli* MG1655 expressing a
463 fluorescent protein YPet fused to the replisome protein DnaN. The cells were grown in MOPS minimal medium + glycerol
464 and 11 amino acids. The dataset was analyzed end-to-end starting from the raw .nd2 file with BACMMAN, DeLTA, and
465 MM3. For analysis with DeLTA, we used the provided channel detection and tracking models but trained a new model on
466 our own data for segmentation. For segmentation with BACMMAN, we used the standard non-learning phase contrast
467 segmentation method ‘MicrochannelPhase2D’. Postprocessing of the output of each pipeline was done in Python. For
468 each pipeline, we filtered for cells whose mothers and daughters were also tracked.

469 The code and data to reproduce the plots in Figure 4 are available at [37] and [44], respectively.

470 For the comparison of Otsu and U-Net outputs from Omnipose in Figure 5, we trained Omnipose with a learning rate of
471 .01 without a pre-trained model. We used the same set of 1000 randomly selected images for both Otsu and U-Net, the
472 only difference coming from the labeled masks themselves. Both models were trained until the loss dipped below 0.9
473 (390 epochs for U-Net, 210 epochs for Otsu). In some cases, the model “hallucinated” cells along the channel features.
474 We excluded these images from the final analysis.

475 Analysis of external datasets

476 The external datasets were preprocessed as follows: BACMMAN, MoMa and molyso datasets were rotated 1-2 degrees
477 to align the channels vertically. BACMMAN, molyso and DeLTA datasets were cropped to remove imaging artifacts from
478 the main trench.

479 The segmentation accuracy of napari-MM3 was quantified as follows: we computed the Jaccard Index as the ratio of
480 true positives (correctly identified cells) to the sum of true positives, false positives (identified cells which were not
481 present in the ground truth data) and false negatives (ground truth cells which were not identified by the segmentation).
482 The segmentation and ground truth masks were determined to be matching if their Intersection over Union value was at
483 least 0.6 (note that two masks become indistinguishable to the human eye at IoU 0.8 and higher [18,47]).

484 The output JSON file and kymographs showing reconstructed cell lineages from each sample datasets are available at
485 [44], along with JSON files containing the parameter values used for each step of the analyses.

486 **U-Net model training**

487 Training data was augmented as described below to aid the generalizability of the model. We trained the U-Net model
488 using a pixel-wise weighted binary cross-entropy loss function, as implemented in DeLTA [22]. The model was trained
489 using the Adam optimizer with a learning rate of 10^{-4} , a dropout rate of 50%, a batch size of 8 samples, a patience (early
490 stopping value) of 50 epochs and a train-test split of 90-10.

491 **Overview of the MM3 pipeline**

492 **Channel compilation and designation**

493 The first section of the MM3 pipeline takes in raw micrographs and returns image stacks corresponding to one growth
494 channel over time. Further pipeline operations are then applied to these stacks.

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

498 All images from a particular FOV are analyzed for the location of channels using the phase contrast plane. Channel
499 detection is performed using a wavelet transform, in which a mask is made which is applied across all time points.
500 Channels are cropped through time using the masks and saved as unique image stacks that include all time points for a
501 given channel and imaging plane. MM3 saves channel stacks in TIFF format.

502 MM3 attempts to compile all channels. However, not all channels contain cells, and some channels may have
503 undesirable artifacts from the device preparation. It is, therefore, desirable to only process certain channels for analysis.
504 Consequently, MM3 auto-detects empty and full channels based on the time correlation of the y-profile of the channel
505 (empty channels are highly correlated in time, while channels containing cells are not). The autodetected channels and
506 their classifications are then displayed in the napari viewer for the user to inspect and modify as needed. The user may
507 also manually select empty channels free of artifacts to be used as templates for phase or fluorescence background
508 subtraction.

509 **Background subtraction**

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

518 **Cell segmentation**

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

522 MM3 has two methods for segmentation: a "standard" method and a supervised learning method. The standard method
523 uses traditional image analysis techniques, specifically background subtraction, Otsu thresholding, morphological
524 operations, and watershed algorithms. As the standard method may require fine-tuning of parameters, the napari plugin
525 allows the user to quickly preview the effect of tuning morphological parameters and threshold value on the
526 segmentation output, without having to process the entire dataset. The Otsu segmentation method first aligns the
527 channel of interest with an empty background channel by computing the orientation, which maximizes the pixel-wise
528 cross-correlation. The empty channel is then subtracted from the full channel, and the image is inverted. Otsu's method
529 is then applied to find the binary threshold value which maximizes the inter-region variance (or equivalently, minimizes
530 the intra-region variance). We then apply a Euclidean distance transform, in which each pixel is labeled with its distance
531 to the dark region. The image is thresholded again, and a morphological opening is applied to erode links between
532 regions. Small objects and objects touching the image border are removed. Each region is labeled, and the labels are
533 used to seed a random walker algorithm [34] on the original image. As implemented in MM3, this "standard" method has

534 three adjustable parameters: the first opening pixel size, second opening pixel size, distance threshold (i.e. threshold
535 which is applied to the distance transformed image, in pixels) and a dimensionless parameter to rescale the
536 Otsu-determined threshold, if needed.

537 The supervised learning method uses a standard U-Net architecture with five levels [21]. The model outputs a cell class
538 probability between 0 and 1 for each pixel, which is thresholded at 0.5 to obtain a binary segmentation. The napari
539 viewer can be used to construct training data, with the option to import existing Otsu or U-Net segmentation output as a
540 template. The neural net can then be trained using a separate widget, with the option to check the performance of the
541 model in the napari viewer after successive rounds of training. We found that applying a weighted loss depending on
542 pixel location - as suggested in the original U-Net paper [21] and implemented for instance in DeLTA [22] - sped up
543 model training and improved segmentation and tracking. Since the accurate separation of adjacent cells is vital for cell
544 tracking, the cost of misidentifying pixels between bordering cells is high. We initially implemented a simple binary
545 weight map where pixels between cells were weighted highly and all others pixels relatively lower. We later added a more
546 complex mapping, drawing directly from the one implemented in DeLTA [12], where weights are maximized on the
547 skeletons [61] of the cells and borders. Intuitively, this weighting tells the model that pixels in the center of the cell, in
548 regions far from cells, and on the borders between cells are most important to predict accurately.

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

554 **Cell tracking**

555 Tracking segmented cells is the second major task in the pipeline. Tracking involves linking cell segments in time in order
556 to define a lineage of cell objects. The default tracking method is a simple decision tree based on *a priori* knowledge of
557 binary fission and the mother machine. For example, cells normally grow by a small amount between time intervals,
558 divide into two similarly sized daughter cells, and cannot pass each other in the channel. The tracking method accounts
559 for the absolute positions and relative ordering of cells in each channel over time. Specifically, at each time point we
560 iterate over all detected regions (potential cells). Based on their relative y positions in the channel and sizes, each is
561 linked to a set of potential descendants / ancestors. When two cells are best matched to the same region, the event is
562 classified as a division, subject to constraints on the size of the regions. This tracking implementation is similar to that
563 employed by BACMMAN [15] although it does not explicitly take into account relative ordering of cells in the channel. It
564 contrasts with more complex optimization-based methods used by other mother machine software[13][13,41].

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

567 **Data output and analysis**

568 Tracking produces a dictionary of cell objects which contains relevant information derived from the cell segments. This
569 includes, but is not limited to, birth and division size, growth rate, and generation time. Each object is identified by a key
570 that represents the FOV and channel of the cell, the time point of its birth, and its position in the channel. Since each cell
571 object has the requisite information to find its corresponding position in the channel stacks, the objects can be modified
572 and extended by additional analysis. For example, the corresponding location of a cell in a fluorescent image stack can
573 be retrieved, focus detection performed, and that information can be added to the cell object. This minimizes the burden
574 of rerunning previous sections of the pipeline for new sub-analyses.

575 Plotting can be done from this cell object dictionary directly, or it can first be converted to a .csv, a pandas DataFrame,
576 or a MATLAB structure. We provide a Jupyter notebook [37] to illustrate how the data can be extracted and plotted.

577 **Fluorescence analysis**

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

580 **Focus tracking**

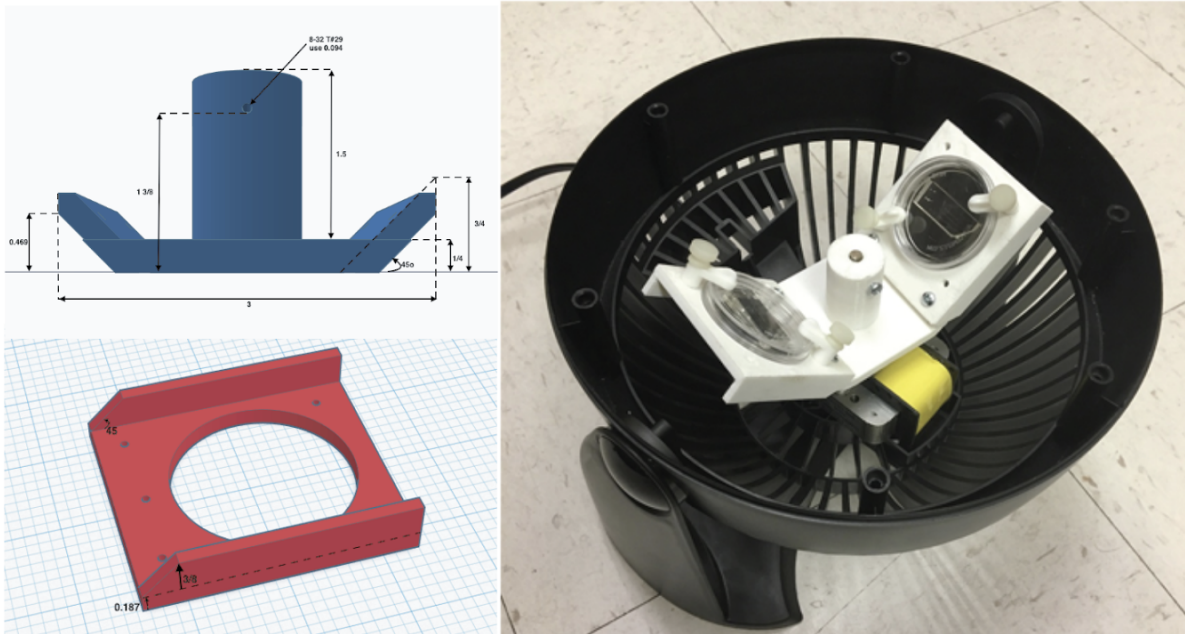
581 The focus tracking module enables the identification and tracking of fluorescent spots or ‘foci.’ This module has been
582 used in our lab for tracking fluorescently labeled replisome machinery in bacteria in order to measure the timing and
583 synchrony of DNA replication initiation. However, it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

586 **U-Net training data annotation**

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

590 Figure S1. Inexpensive fabrication of cell loader with 3D printing.

591 An inexpensive device for loading cells into the mother machine. The construction involves 3D printing a custom holder/
592 rotor for a 50mm WillCo dish, on which a mother machine is attached. The holder is printed in three parts (2 blades and a
593 central base) to account for 3D printers with small printing areas. This piece is then assembled and secured to a
594 Honeywell fan from which the original blade has been removed. CAD files and details of the fan centrifuge construction
595 are available at [28].



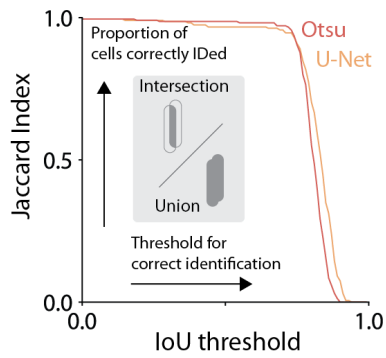
596

597

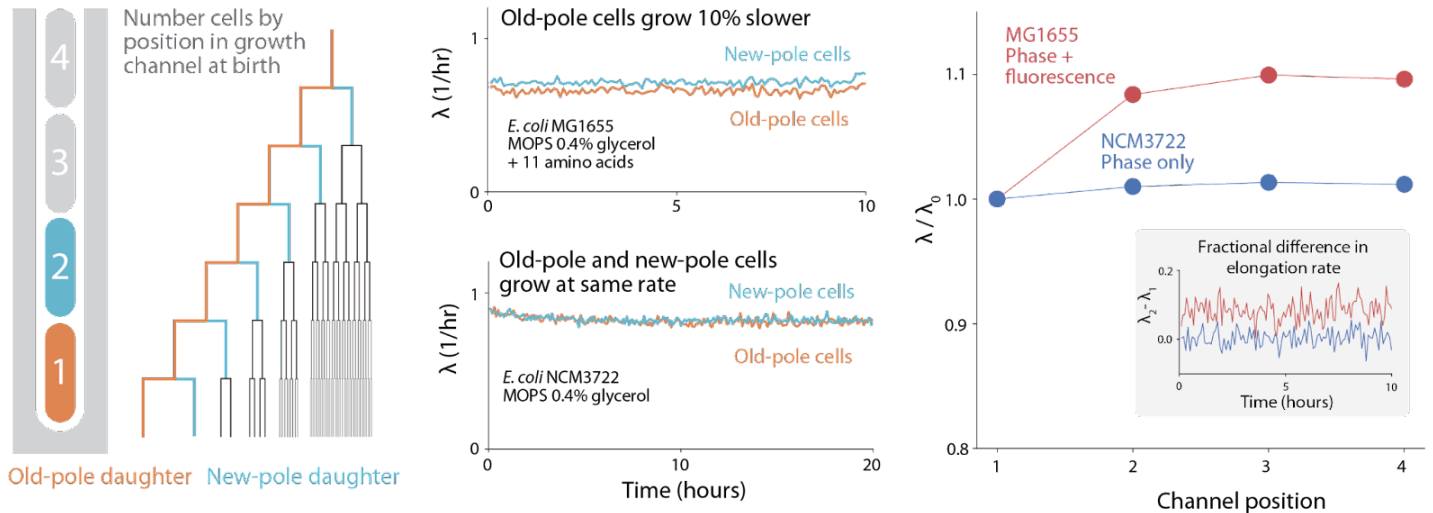
598 Figure S2: Segmentation accuracy of napari-MM3 Otsu and U-Net methods

599 To quantify the accuracy of the segmentation masks generated by MM3's Otsu and U-Net segmentation methods, we
600 computed the Jaccard Index [47,50] as a function of the intersection-over-union (IoU) threshold.

601



602 **Figure S3: Old-pole aging phenotype is strain specific.** Cells imaged with fluorescence often show signs of aging in the
603 old-pole “mother” cell. For instance, in the dataset analyzed in Figure 4 (*E. coli* MG1655 with the fluorescent protein YPet fused
604 to DnaN), we observed systematic differences in cell elongation rate and size between the old-pole cell at the end of the growth
605 channel and its sisters, which inherit the new pole (top center). However, this asymmetry is not universal. Using napari-MM3’s
606 Otsu segmentation method, we re-analyzed previously published data obtained without fluorescence illumination [32], and found
607 that the old-pole and new-pole cell elongation rates varied only on the order of 1% (lower center), while in the dataset obtained
608 under fluorescence imaging, the old-pole mother cells grow 7-10% slower than the new pole cells. These results are consistent
609 with a previous survey [62], which found that most evidence for aging in *E. coli* comes from studies utilizing fluorescent proteins
610 for visualization.



611

612

613 References

- 614 1. Wang P, Robert L, Pelletier J, Dang WL, Taddei F, Wright A, et al. Robust growth of *Escherichia coli*. *Curr Biol*.
615 2010;20: 1099–1103.
- 616 2. Sauls JT, Cox SE, Do Q, Castillo V, Ghulam-Jelani Z, Jun S. Control of *Bacillus subtilis* Replication Initiation during
617 Physiological Transitions and Perturbations. *MBio*. 2019;10. doi:10.1128/mBio.02205-19
- 618 3. Nakaoka H, Wakamoto Y. Aging, mortality, and the fast growth trade-off of *Schizosaccharomyces pombe*. *PLoS*
619 *Biol*. 2017;15: 1–29.
- 620 4. Spivey EC, Jones SK, Rybarski JR, Saifuddin FA, Finkelstein IJ. An aging-independent replicative lifespan in a
621 symmetrically dividing eukaryote. *Elife*. 2017;6: 1–25.
- 622 5. Jun S, Si F, Pugatch R, Scott M. Fundamental principles in bacterial physiology—history, recent progress, and the
623 future with focus on cell size control: a review. *Rep Prog Phys*. 2018;81: 056601.
- 624 6. Bakshi S, Leoncini E, Baker C, Cañas-Duarte SJ, Okumus B, Paulsson J. Tracking bacterial lineages in complex and
625 dynamic environments with applications for growth control and persistence. *Nat Microbiol*. 2021;6: 783–791.
- 626 7. Kaplan Y, Reich S, Oster E, Maoz S, Levin-Reisman I, Ronin I, et al. Observation of universal ageing dynamics in
627 antibiotic persistence. *Nature*. 2021;600: 290–294.
- 628 8. Russell JR, Cabeen MT, Wiggins PA, Paulsson J, Losick R. Noise in a phosphorelay drives stochastic entry into
629 sporulation in *Bacillus subtilis*. *EMBO J*. 2017;36: 2856–2869.
- 630 9. Amir A, Babaeipour F, McIntosh DB, Nelson DR, Jun S. Bending forces plastically deform growing bacterial cell
631 walls. *Proc Natl Acad Sci U S A*. 2014;111: 5778–5783.
- 632 10. Si F, Le Treut G, Sauls JT, Vadia S, Levin PA, Jun S. Mechanistic Origin of Cell-Size Control and Homeostasis in
633 Bacteria. *Curr Biol*. 2019;29: 1760–1770.e7.
- 634 11. Stewart EJ, Madden R, Paul G, Taddei F. Aging and death in an organism that reproduces by morphologically
635 symmetric division. *PLoS Biol*. 2005;3: e45.
- 636 12. O'Connor OM, Alnahhas RN, Lugagne J-B, Dunlop MJ. DeLTA 2.0: A deep learning pipeline for quantifying
637 single-cell spatial and temporal dynamics. *PLoS Comput Biol*. 2022;18: e1009797.
- 638 13. Sachs CC, Grünberger A, Helfrich S, Probst C, Wiechert W, Kohlheyer D, et al. Image-Based Single Cell Profiling:
639 High-Throughput Processing of Mother Machine Experiments. *PLoS One*. 2016;11: e0163453.
- 640 14. Smith A, Metz J, Pagliara S. MMHelper: An automated framework for the analysis of microscopy images acquired
641 with the mother machine. *Sci Rep*. 2019;9: 10123.
- 642 15. Ollion J, Elez M, Robert L. High-throughput detection and tracking of cells and intracellular spots in mother machine
643 experiments. *Nat Protoc*. 2019;14: 3144–3161.
- 644 16. Stylianidou S, Brennan C, Nissen SB, Kuwada NJ, Wiggins PA. SuperSegger: robust image segmentation, analysis
645 and lineage tracking of bacterial cells. *Mol Microbiol*. 2016;102: 690–700.
- 646 17. Panigrahi S, Murat D, Le Gall A, Martineau E, Goldlust K, Fiche J-B, et al. Misic, a general deep learning-based
647 method for the high-throughput cell segmentation of complex bacterial communities. *Elife*. 2021;10.
648 doi:10.7554/eLife.65151
- 649 18. Cutler KJ, Stringer C, Wiggins PA, Mougous JD. Omnipose: a high-precision morphology-independent solution for
650 bacterial cell segmentation. *Nat Methods*. 2022; 2021.11.03.467199.

- 651 19. Spahn C, Gómez-de-Mariscal E, Laine RF, Pereira PM, von Chamier L, Conduit M, et al. DeepBacs for multi-task
652 bacterial image analysis using open-source deep learning approaches. *Commun Biol.* 2022;5: 688.
- 653 20. Moen E, Borba E, Miller G, Schwartz M, Bannon D, Koe N, et al. Accurate cell tracking and lineage construction in
654 live-cell imaging experiments with deep learning. *bioRxiv.* 2019. p. 803205. doi:10.1101/803205
- 655 21. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. *Lect Notes*
656 *Comput Sci.* 2015;9351: 234–241.
- 657 22. Lugagne JB, Lin H, Dunlop MJ. Delta: Automated cell segmentation, tracking, and lineage reconstruction using deep
658 learning. *PLoS Comput Biol.* 2020;16: 1–18.
- 659 23. Taheri-Araghi S, Brown SD, Sauls JT, McIntosh DB, Jun S. Single-Cell Physiology. *Annu Rev Biophys.* 2015;44:
660 123–142.
- 661 24. Allard P, Papazotos F, Potvin-Trottier L. Microfluidics for long-term single-cell time-lapse microscopy: Advances and
662 applications. *Front Bioeng Biotechnol.* 2022;10: 968342.
- 663 25. Potvin-Trottier L, Luro S, Paulsson J. Microfluidics and single-cell microscopy to study stochastic processes in
664 bacteria. *Curr Opin Microbiol.* 2018;43: 186–192.
- 665 26. Cabeen MT, Losick R. Single-cell Microfluidic Analysis of *Bacillus subtilis*. *J Vis Exp.* 2018. doi:10.3791/56901
- 666 27. Kamande JW, Wang Y, Taylor AM. Cloning SU8 silicon masters using epoxy resins to increase feature replicability
667 and production for cell culture devices. *Biomicrofluidics.* 2015;9: 036502.
- 668 28. mother-machine-protocols: Procedures for duplicating, constructing and using the microfluidic mother machine
669 device. Github; Available: <https://github.com/junlabucsd/mother-machine-protocols>
- 670 29. Napari hub. [cited 17 Jan 2023]. Available: <https://www.napari-hub.org/plugins/napari-mm3>
- 671 30. napari — napari. [cited 23 Jan 2023]. Available: <https://napari.org/stable/>
- 672 31. Rideau F, Villa A, Belzanne P, Verdier E, Hosy E, Arfi Y. Imaging Minimal Bacteria at the Nanoscale: a Reliable and
673 Versatile Process to Perform Single-Molecule Localization Microscopy in Mycoplasmas. *Microbiol Spectr.* 2022;10:
674 e0064522.
- 675 32. Taheri-Araghi S, Bradde S, Sauls JT, Hill NS, Levin PA, Paulsson J, et al. Cell-size control and homeostasis in
676 bacteria. *Curr Biol.* 2015;25: 385–391.
- 677 33. Sauls JT, Schroeder JW, Si F, Brown SD, Le Treut G, Wang JD. Mother machine image analysis with MM3. *bioRxiv.*
678 2019; 4–7.
- 679 34. Grady L. Random walks for image segmentation. *IEEE Trans Pattern Anal Mach Intell.* 2006;28: 1768–1783.
- 680 35. Falk T, Mai D, Bensch R, Çiçek Ö, Abdulkadir A, Marrakchi Y, et al. U-Net: deep learning for cell counting, detection,
681 and morphometry. *Nat Methods.* 2019;16: 67–70.
- 682 36. Otsu N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans Syst Man Cybern.* 1979;9: 62–66.
- 683 37. notebooks at main · junlabucsd/napari-mm3. Github; Available: <https://github.com/junlabucsd/napari-mm3>
- 684 38. Ollion J, Ollion C. DistNet: Deep Tracking by Displacement Regression: Application to Bacteria Growing in the
685 Mother Machine. *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020.* Springer
686 International Publishing; 2020. pp. 215–225.
- 687 39. Apple Inc. Tensorflow plugin - metal. In: Apple Developer [Internet]. [cited 14 Jul 2023]. Available:
688 <https://developer.apple.com/metal/tensorflow-plugin/>

- 689 40. Banerjee DS, Stephenson G, Das SG. Segmentation and analysis of mother machine data: SAM. *bioRxiv*. 2020. p.
690 2020.10.01.322685. doi:10.1101/2020.10.01.322685
- 691 41. Kaiser M, Jug F, Julou T, Deshpande S, Pfohl T, Silander OK, et al. Monitoring single-cell gene regulation under
692 dynamically controllable conditions with integrated microfluidics and software. *Nat Commun*. 2018;9.
693 doi:10.1038/s41467-017-02505-0
- 694 42. Stringer C, Wang T, Michaelos M, Pachitariu M. Cellpose: a generalist algorithm for cellular segmentation. *Nat*
695 *Methods*. 2021;18: 100–106.
- 696 43. Ollion J. bacmman. Github; Available: <https://github.com/jeanollion/bacmman>
- 697 44. mother-machine-data: A repository for processed mother machine data from the Jun Lab. Github; Available:
698 <https://github.com/junlabucsd/mother-machine-data>
- 699 45. Le Treut G, Si F, Li D, Jun S. Quantitative Examination of Five Stochastic Cell-Cycle and Cell-Size Control Models for
700 *Escherichia coli* and *Bacillus subtilis*. *Front Microbiol*. 2021;12: 721899.
- 701 46. Geiger RS, Cope D, Ip J, Lotosh M, Shah A, Weng J, et al. “Garbage in, garbage out” revisited: What do machine
702 learning application papers report about human-labeled training data? *Quant Sci Stud*. 2021;2: 795–827.
- 703 47. Laine RF, Arganda-Carreras I, Henriques R, Jacquemet G. Avoiding a replication crisis in deep-learning-based
704 bioimage analysis. *Nat Methods*. 2021;18: 1136–1144.
- 705 48. Babbage C. *Passages from the life of a philosopher*. Theclassics; 2013.
- 706 49. Mellin WD. Work with new electronic “brains” opens field for army math experts. *Hammond Times*.
- 707 50. Jeckel H, Drescher K. Advances and opportunities in image analysis of bacterial cells and communities. *FEMS*
708 *Microbiol Rev*. 2021;45. doi:10.1093/femsre/fuaa062
- 709 51. Image.Sc forum. In: Image.sc Forum [Internet]. [cited 24 Jan 2023]. Available: <https://forum.image.sc/>
- 710 52. Bourne R. ImageJ. In: Bourne R, editor. *Fundamentals of Digital Imaging in Medicine*. London: Springer London;
711 2010. pp. 185–188.
- 712 53. Assets — DeLTA 2.0-gamma documentation. [cited 23 Feb 2023]. Available:
713 https://delta.readthedocs.io/en/latest/usage/assets_desc.html
- 714 54. Shiaelis N, Tometzki A, Peto L, McMahon A, Hepp C, Bickerton E, et al. Virus detection and identification in minutes
715 using single-particle imaging and deep learning. *ACS Nano*. 2023;17: 697–710.
- 716 55. Hardo G, Noka M, Bakshi S. Synthetic Micrographs of Bacteria (SyMBac) allows accurate segmentation of bacterial
717 cells using deep neural networks. *BMC Biol*. 2022;20: 263.
- 718 56. Naylor P, Lae M, Reyal F, Walter T. Segmentation of Nuclei in Histopathology Images by Deep Regression of the
719 Distance Map. *IEEE Trans Med Imaging*. 2019;38: 448–459.
- 720 57. Lawson MJ, Camsund D, Larsson J, Baltekin Ö, Fange D, Elf J. In situ genotyping of a pooled strain library after
721 characterizing complex phenotypes. *Mol Syst Biol*. 2017;13: 947.
- 722 58. Luro S, Potvin-Trottier L, Okumus B, Paulsson J. Isolating live cells after high-throughput, long-term, time-lapse
723 microscopy. *Nature Methods*. 2020. pp. 93–100. doi:10.1038/s41592-019-0620-7
- 724 59. Microfluidic Device Analyzer napari Plugin. In: Chan Zuckerberg Initiative [Internet]. 11 Nov 2021 [cited 26 Jul 2023].
725 Available:
726 <https://chanzuckerberg.com/science/programs-resources/imaging/napari/microfluidic-device-analyzer-napari-plugin/>
727 /

- 728 60. napari-mm3: Mother machine image analysis through napari. Github; Available:
729 <https://github.com/junlabucsd/napari-mm3>
- 730 61. Lee TC, Kashyap RL, Chu CN. Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms. CVGIP:
731 Graphical Models and Image Processing. 1994;56: 462–478.
- 732 62. Rang CU, Peng AY, Poon AF, Chao L. Ageing in Escherichia coli requires damage by an extrinsic agent.
733 Microbiology. 2012;158: 1553–1559.
- 734