Utilizing modern machine learning approaches for image cytometry

Kevin Cutler

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2023

Reading Committee:

Joseph Mougous, Chair

Paul Wiggins, Chair

Armita Nourmohammad

Program Authorized to Offer Degree:

Physics

University of Washington

Abstract

Utilizing modern machine learning approaches
for image cytometry

Kevin Cutler

Co-Chairs of the Supervisory Committee:

Professor Joseph Mougous

Microbiology

Professor Paul Wiggins

Physics

Until recently, the scientific community has lacked image segmentation tools that are precise, reliable, and general-purpose. Such tools are especially needed in applications to bacterial image cytometry, wherein single-pixel precision is needed, perfect segmentation must be achieved over thousands of cells over hundreds of time points, and the approach must be applicable to a diversity of cellular morphologies present in a single micrograph. In this document, I detail the challenges of bacterial image segmentation, the failures of prior approaches, and the use of machine learning to solve this problem in virtually any unilaminar cell imaging context.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

A22	A cell-permeable isothiourea compound that specifically, rapidly, and reversibly perturbs MreB function without affecting eukaryotic actin polymerization. It affects the growth and morphology of *C. crescentus*, converts rod-shaped *E. coli* to round cells, and blocks DNA segregation without affecting DNA replication [1].	11, 78
A. baylyi	*Acinetobacter baylyi*. A nonmotile, gram negative coccobacillus that grows under aerobic conditions.	13, 14
A. thaliana	*Arabidopsis thaliana*. A small plant from the mustard family (Brassicaceae), native to Eurasia and Africa, commonly found along the shoulders of roads and in disturbed land.	31, 35
aztreonam	An antibiotic used primarily to treat infections caused by gram-negative bacteria.	20, 78
B. subtilis	*Bacillus subtilis*. A bacterium found in soil and the gastrointestinal tract of ruminants and humans, often used in biotechnology.	13, 16, 28
B. thailandensis	*Burkholderia thailandensis*. A Gram-negative, rod-shaped bacterium that naturally occurs in soil. It is closely related to *Burkholderia pseudomallei*, but unlike *B. pseudomallei*, it only rarely causes disease in humans or animals.	13
C. crescentus	*Caulobacter crescentus*. A Gram-negative, oligotrophic bacterium widely distributed in fresh water lakes and streams.	13, 20, 28, 78
C. elegans	*Caenorhabditis elegans*. A nematode that is a model organism in biology, known for its transparent skin and the simplicity of its nervous system.	75

cephalexin	A beta-lactam antibiotic within the class of first-generation cephalosporins. It kills gram-positive and some gram-negative bacteria by disrupting the growth of the bacterial cell wall.	29, 49, 78
consensus image	An image created by registering and averaging signal from populations of cells at the same stage in their life cycle. This is done to improve the signal-to-noise ratio and reveal patterns in living cells.	51
DNN	Deep Neural Network. A type of neural network with many layers. A universal function approximator.	7, 82
E. coli	*Escherichia coli*. A common bacterium found in the gut of warm-blooded organisms, with some strains causing illnesses.	1, 13, 16, 20, 79
F. tularensis subsp *novicida*	*Francisella tularensis* subsp *novicida*. A subspecies of *Francisella tularensis* that is highly virulent in mice but has little to no effect on humans.	13
GUI	Graphical User Interface. A type of interface that allows users to interact with a program using graphical elements such as windows, buttons, and menus.	50, 63
H. pylori	*Helicobacter pylori*. A bacterium that can cause stomach ulcers and is often found in the stomach.	13, 20, 78
kymograph	A type of image that is created by taking a line of pixels from each frame of a video and stacking them on top of each other.	51
lineage tree	A tree that shows the lineage of a cell. In biology, this is often used to show the lineage of a cell from its parent cell to its daughter cells.	56

modality	Modality refers to the type of imaging technique used to acquire the image data, such as phase contrast, DIC, and epifluorescnece, among many others.	10
P. aeruginosa	*Pseudomonas aeruginosa*. A bacterium that can cause disease in plants and animals, including humans, and is known for its antibiotic resistance.	13, 16, 28
photobleaching	The loss of fluorescence signal due to the excitation of fluorophores.	62
S. aureus	*Staphylococcus aureus*. A bacterium that can cause various infections, often found on the skin and in the respiratory tract.	13, 16, 28
S. flexneri	*Shigella flexneri*. A species of Gram-negative bacteria in the genus Shigella that can cause diarrhea in humans.	13, 28
S. proteamaculans	*Serratia proteamaculans*. A Gram-negative, facultatively anaerobic, rod-shaped bacterium.	13, 79
S. pristinaespiralis	*Streptomyces pristinaespiralis*. A bacterium known for its role in the biosynthesis of pristinamycin, a clinically important antibiotic.	13, 16, 28, 79
U-Net	A type of neural network architecture that is commonly used for image segmentation. It is composed of a contracting path and an expansive path. The contracting path is composed of convolutional layers and max pooling layers, and the expansive path is composed of convolutional layers and upsampling layers. Skip connections are used to connect the contracting path to the expansive path.	7, 11
V. cholerae	*Vibrio cholerae*. A bacterium that causes cholera, a severe diarrheal disease, often found in contaminated water.	13, 16, 28

ACKNOWLEDGMENTS

I first must give great appreciation to Peter Schaffer, Paula Heron, Donna Messina, and the rest of the Physics Education Group. Although I was ultimately drawn to biology and computer science, I continue to admire their work and am grateful for being part of their group in my early graduate years.

Special thanks must be given to the late Lillian McDermott, whom I met through her Physics by Inquiry course (then taught by Donna) in my last year of undergraduate studies at UW. We worked almost daily on her book for several years and she became one of my closest friends.

My colleagues in the Wiggins and Mougous labs have been invaluable. Kaitlin and See-Yeun taught me a good bit of biology early on, and See-Yeun helped us make some invaluable fluorescent fusions in *B. thailandensis*. Through dozens of lab meetings and practice talks, the Mougous lab transformed my ability to communicate science effectively. Meanwhile, Paul and Isaac kept me sharp on optics, and the rest of my friends in the Wiggins lab kept me sane.

I must also extend my deep gratitude to Paul Wiggins, Joseph Mougous, and S. Brook Peterson. The time and effort they invested into my research and professional development was immense and truly appreciated.

Of course, I must also acknowledge my funding. I love teaching, but a TA grinds research to a halt. It was only with the time afforded by my RA with Joseph and the Molecular Biophysics Training Grant (which Joseph and Brook very helpfully pushed me to get) that I was able to produce Omnipose and my associated work.

Help from collaborators was essential to the development of Omnipose and the associated training data. Carsen Stringer, developer of Cellpose, did a lot of work helping me to inte-

grate Omnipose into the main Cellpose branch on its initial release. Luca Rappez contributed both images and ground truth for high-resolution *C. elegans* training data. Dave Kysela in the Brun lab contributed images of *C. crescentus*. Sophie Sichel in the Salama lab contributed fixed and stained *H. pylori* for me to image. Andrea Vettiger in the Bernhardt lab sent me Pal-mCherry *E. coli*, which I used for multiple ground truth datasets (most notably, spacetime segmentation). Lastly, Joanna Timmins and Jean-Philippe Kleman contributed superb high-frame-rate multichannel confocal data of Nile-Red-stained *D. radiodurans* cells for use in training self-contact, spacetime, and multimodal models.

Lastly, I would like to express my appreciation to all the people that made this possible. The administration in the Microbiology and Physics departments, particularly Catherine Provost and Dima Young, patiently supported me through this cross-departmental research arrangement. I also could not have found better committee members than Beth Traxler, Dan Fu, Miguel Morales, Armita Nourmohammad, Paul, and Joseph. And of course, I am immensely thankful for the love and support of my family.

Chapter 1

INTRODUCTION

Although light microscopy is a valuable tool for characterizing cellular and sub-cellular structures and dynamics, quantitative analysis of microscopy images remains a persistent challenge [2]. This is especially pertinent to the study of bacteria, many of which have dimensions in the range of visible wavelengths. Thus, a typical bacterial cell body is composed of a small number of pixels (∼100–300 px^2 for *E. coli* in typical experiments). At this scale, accurate subcellular localization requires defining the cell boundary with single-pixel precision. The process of defining boundaries within images is termed segmentation, and this is a critical first step in image analysis pipelines [3, 4].

1.1 Strategies for encoding image segmentation

Depending on the level of detail needed to represent segmented objects, we may choose from three modes of segmentation: semantic, instance, and affinity (Fig. 1.1).

1.1.1 Semantic labels

The lowest-information approach to image segmentation is called *semantic segmentation*, which sorts pixels into semantic classes. This is often just two classes, *i.e.*, binary classification into foreground and background. In this context, foreground is cell and background is media. Binary segmentation results are stored as a binary image.

Semantic segmentation does not discern between adjacent instances of foreground objects (Fig. 1.1b). Accordingly, only boundaries between foreground and background are encoded by binary semantic labels.

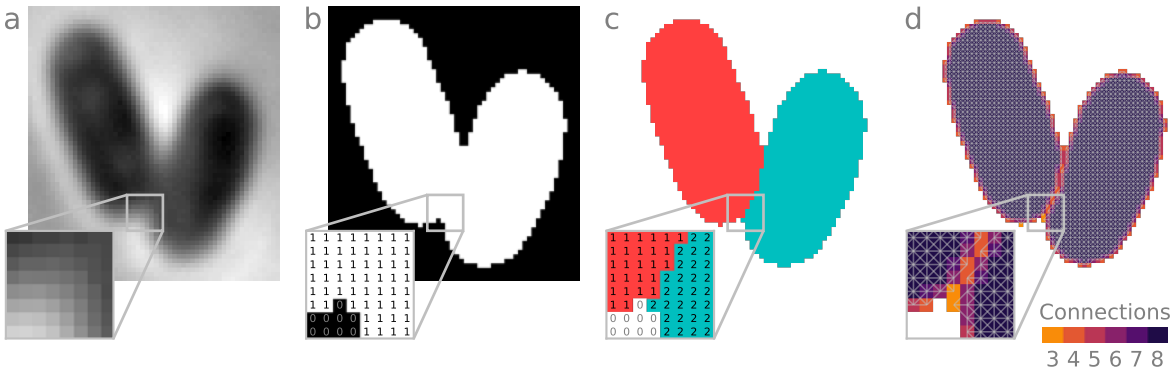

Figure 1.1: Three distinct classes of segmentation encoding. (a) Typical phase contrast image of *E. coli* cells. (b) Semantic segmentaion. (c) Instance segmentation. (d) Affinity segmentation.

We store a semantic segmentation as a binary image file with the same dimensions as the image itself, with foreground pixels labeled `True` (1) and background `False` (0).

1.1.2 Instance labels

Instance segmentation assigns a unique integer to the pixels each instance of an object - in this case, each cell (Fig. 1.1c). These labels, also commonly referred to as a *label matrix*, is likewise stored as an image file, typically uint8 (unsigned 8-bit integer) for up to 255 ($2^8 - 1$) labels or uint16 (unsigned 16-bit integer) for up to 65535 ($2^{16} - 1$) labels.*

Instance labels implicitly define cell boundaries between instances. These boundaries can be computed by comparing adjacent pixels. Pixel pairs with dissimilar labels are considered boundary pixels. To obtain a 2-connected boundary, pairs in the cardinal and ordinal directions are considered. To obtain a 1-connected boundary, only pairs in cardinal directions are considered.

*Signed and/or unsigned 32- or 64-bit formats may also be used, but some operating systems may not be able to preview these files in their native file managers

Instance labels are easy to visualize, edit, and store. An instance label matrix is an 8- or 16-bit image of the same dimensions as the source image.* Instance labels are visualized as distinct color overlays or boundary overlays (or a combination thereof), and multidimensional image viewers like Napari can be used to directly modify labels in these visual modes [5].

1.1.3 Affinity graphs

Despite the convenience and general applicability of instance labels, they cannot be used to completely describe all cells. Specifically, a label matrix for an object exhibiting self-contact morphology has no means of encoding the boundaries of that at the points of self-contact (Fig. 1.2).

However, an affinity graph does encode this information. By storing pairwise connectivity for all adjacent pixels, the affinity graph (also known as an *adjacency list*) implicitly stores internal pixels (fully connected) and boundary pixels (not fully connected) (Fig. 1.1d). This graph can be stored as a 2D array, with each column representing the connections of a given pixel (a graph node) and each row representing connectivity to an adjacent neighbor in a given direction (a graph edge).†‡

The affinity graph has more information than just identifying cells and their boundaries; by using a right- (or left-) hand rule, the boundaries can be traversed/parametrized (Fig. 1.3d). This alone is useful for analyzing membrane-bound signal, but in principle, the affinity graph can be used to fully define an internal coordinate system and a coordinate transform to a standard cell shape. This key operation, required to synthesize position-dependent signal

---

*Images with shape `(C,Y,X)`, where `C` is the channel axis, would have an instance label image of shape `(Y,X)`.

†Pixels (or in ND, hypervoxels) may be classified as interior or boundary by their net connectivity. An ND hypervoxel connected to all $3^N - 1$ neighbors is classified as internal (8 in 2D, fully 2-connected to both cardinal and ordinal neighbors). Hypervoxels with fewer than $3^N - 1$ connections are classified as boundary.

‡The affinity graph is thus of shape $(m, 3^N - 1)$ where m is the number of hypervoxels. This is at worst equivalent to $3^N - 1$ binary matrices the same size as the source image, but can be made more efficient by storing only foreground pixels as nodes in the graph.

collected from populations of cells, has not yet been implemented for cells with arbitrary topologies.

Although not all segmentation algorithms can be made to output affinity graphs in addition to instance labels, Omnipose can. This is the subject of Chapter 4.

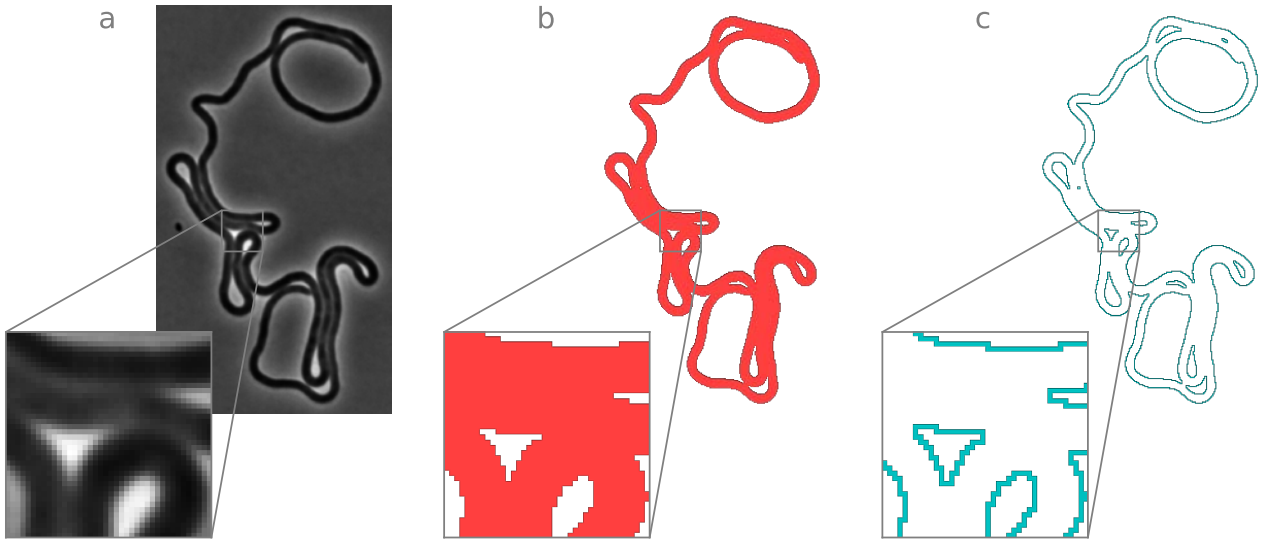

Figure 1.2: Instance labels fail to describe self-contact topology. (a) Phase contrast image of elongaed *E. coli* cell grown on $1\,\mu g\,mL^{-1}$ aztreonam. (b) Cell mask is a uniform label across self contact points. (c) Boundaries cannot be detected at points of self-contact.

1.2 Strategies for achieving image segmentation

Cell segmentation is a complex problem that extends beyond microbiological research, and so many solutions are currently available in image analysis programs [6–24]. Most of these solutions use traditional image processing techniques applied to raw images or DNN (deep neural network) transformations of images to obtain binary semantic labeks that are then converted to instance labels.

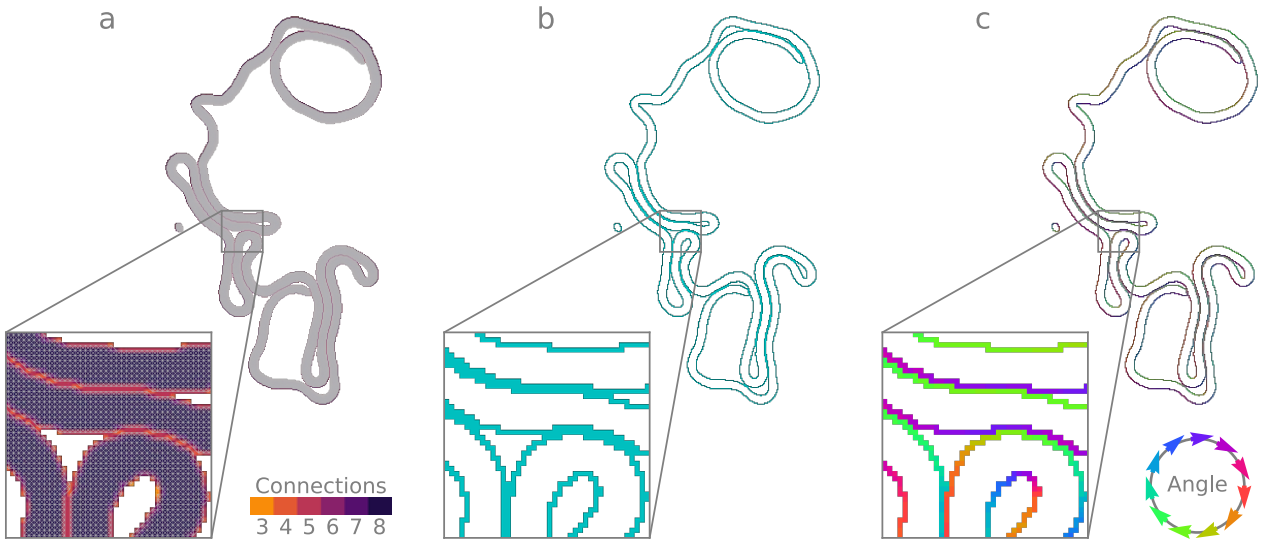

Figure 1.3: Affinity graphs encode rich topological information. (a) Affinity graph for cell in Fig. 1.2a. Color indicates the number of connections to adjacent pixels. Connection between pixels (affinity=1) indicated by white line. (b) Boundary map extracted from the affinity graph. (c) Parametrized cell contour. Color indicates direction along boundary.

1.2.1 Semantic islands to instance labels

Semantic segmentation can be converted into instance segmentation, and this forms the basis of many instance segmentation pipelines. The general steps are:

1. Pre-process image: traditional filtering/blurring/feature extraction or DNN transformation of raw data.

2. Threshold processed image: adaptive techniques are usually used on the pre-processed image to ensure that the majority of objects pixels are identified despite variations within an image and among images in a dataset. Importantly, object boundaries must not be identified as foreground. This allows each object to be associated with a unique island of foreground pixels.

3. Identify unique blobs using connected components labeling. This is the process of building an affinity graph, where pixels are nodes and edges are formed between any adjacent foreground pixels. Adjacency can be defined most narrowly by sharing edges (1-connected in Python, 4-connected in MATLAB) or more broadly by sharing either edges or vertices (2-connected in Python, 8-connected in MATLAB). The graph is then traversed to find all connected components of the graph.

Step (3) is especially problematic, as it requires large gaps between foreground islands to guarantee that cells are not joined together (Fig. 1.4). This poses a challenge to extracting accurate cell size, shape, and signal localized or measured relative to the cell boundary.

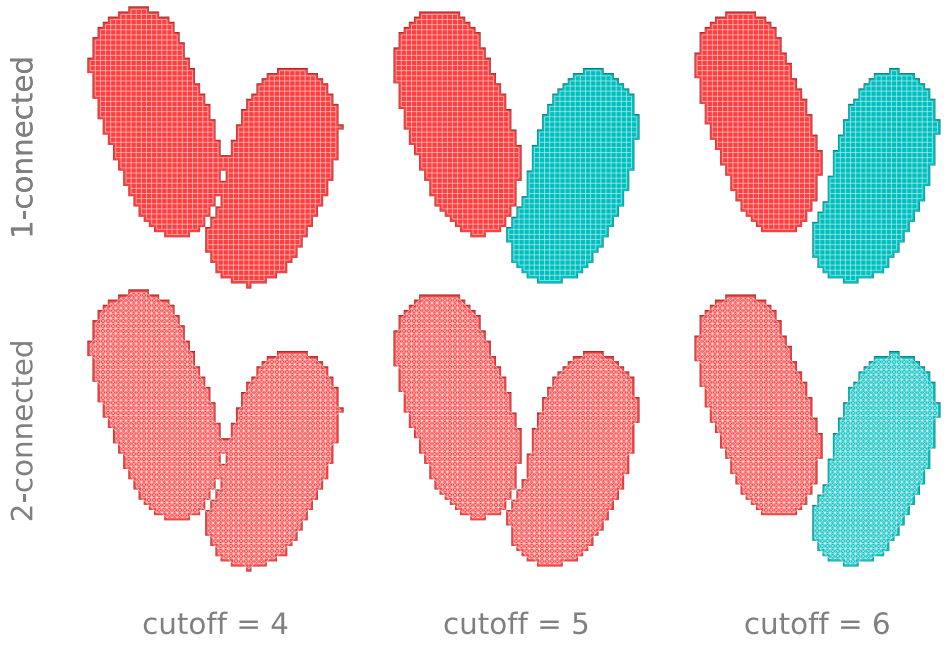

Figure 1.4: Boundaries are often lost by traditional segmentation. By simulating the amount of foreground pixels detected by filtering+thresholding, we see that it is impossible to distinguish between the two cells until much of the boundary is lost, particularly when using 2-connectivity. Pixel connectivity do neighbors represented by white lines.

Missing boundary pixels also frequently arise when applying the watershed transform. As usually implemented, this ubiquitous operation returns a semantic classification of an image into watershed lines (cell boundaries) and catchment basins (cell interiors). This means that distinct basins must be separated by a 1- or 2-connected watershed line, and therefore boundary pixels are always left unclassified.*

SuperSegger was developed to address thresholding and watershed issues specifically in bacterial phase contrast images [10]. This program utilizes traditional image filtering techniques to generate candidate watershed lines and a shallow neural network to reject spurious lines that would lead to over-segmentation. However, missing boundary pixels remain a fundamental limiation of this watershed-based approach.

1.2.2 The promise of deep neural networks

Deep neural networks (DNNs) are now widely recognized as superior tools for cell segmentation [25]. Unlike traditional image processing, machine learning approaches such as DNNs require training on a ground-truth dataset of cells and corresponding labels. Trained DNNs are thus limited in applicability to images that are representative of those in the training dataset. Early DNN approaches were based on the Mask R-CNN architecture [21], whereas more recent algorithms such as StarDist, Cellpose, and MiSiC are based on the U-Net architecture [9, 12, 23].

Pachitariu and colleagues showed that Cellpose outperforms Mask R-CNN and StarDist on a variety of cell types and cell-like objects, distinguishing it as a general solution for cell segmentation [9]. Notably, the representation of bacteria in their study was limited. MiSiC was developed as a general DNN-based solution for bacterial segmentation; however, the

*There are implementations that allow users to return instance labels without the gaps let by watershed lines (*e.g.*, `skimage.segmentation.watershed()`), but such implementations are not widely adopted. Despite this fix, watershed also tends to over-segment images (even when transformed by traditional filters or DNNs).

authors of MiSiC did not provide comparisons to other DNN algorithms [12]. In Chapter 2, we evaluate the performance of state-of-the-art cell segmentation algorithms on a diverse collection of bacterial cells.

1.2.3 Mask reconstruction

DNNs do not directly predict instance labels. This is fundamentally because instance labels are effectively random numerical labels, but a neural network can only learn to approximate well-defined (non-random) numbers. Additionally, neural networks always predict smoothed/interpolated output relative to the training data, particularly at cell edges. This is not suitable for instance labels, which must be contiguous and integer-valued over the entire extent of the cell.

In all cases, DNNs are trained to predict some kind of image transformation that is amenable to processing into semantic, instance, or affinity segmentation. In Chapter 2, I outline the various network outputs and mask reconstruction strategies.

1.3 The challenge of morphology

In addition to their small size, bacteria adopt a wide range of morphologies. Although many commonly studied bacteria are well-approximated by rods or spheres, there is growing interest in bacteria with more elaborate shapes [26]. Some examples include Streptomycetales, which form long filamentous and branched structures [27], and Caulobacterales, which possess extended appendages distinct from their cytoplasm [28]. Furthermore, microfluidic devices are allowing researchers to capture the responses of bacteria to assorted treatments such as antibiotics, which often result in highly irregular morphologies [29]. Whether native or induced, atypical cell morphologies present a distinct problem at the cell segmentation phase of image analysis [6, 7]. This is compounded when such cells are present with those adopting other morphologies, as is the case in many natural samples of interest [30].

Although some algorithms referenced thus far can be trained or otherwise fine-tuned on varied morphologies, we found that none of them constituted a generalizable solution for segmenting bacterial cells of assorted size and shape. This motivated the design of a new algorithm, Omnipose, that significantly outperforms all previous cell segmentation algorithms across a wide range of bacterial cell sizes, morphologies, and optical characteristics.

Chapter 2

PRIOR WORK

Numerous image segmentation algorithms have been developed, and the performance of many of these on bacterial cells is documented [2]. These broadly fall into three categories: (i) traditional image processing approaches (*e.g.*, thresholding, watershed), (ii) traditional/machine learning hybrid approaches, and (iii) deep neural network (DNN) approaches. Given the goal of developing software with the capacity to recognize bacteria universally, I sought to identify strongly performing algorithms for further development. An unbiased, quantitative comparison of cell segmentation algorithms on bacterial cells had not yet been performed; thus, I selected one or more representatives from each category for this analysis: (i) Morphometrics [20], (ii) SuperSegger [10], (iii) Mask R-CNN [24], StarDist [23], MiSiC [12], and Cellpose [9].

2.1 Rationale behind the selection of segmentation algorithms

Three main factors contributed to the choice of algorithms highlighted here: (i) specificity to bacterial phase contrast images, (ii) success and community adoption, especially for bioimage segmentation, and (iii) feasibility of installation, training, and use. It is important to note that criterion (i) only influenced the choice of non-DNN algorithms because they are generally modality-, scale-, and subject-specific in their design. DNN approaches can generally be trained on arbitrary sets of images. With the exception of MiSiC, none of the DNN-based approaches we chose were specifically designed for (or substantively trained on) bacterial phase contrast images.

SuperSegger, Morphometrics, and MiSiC were selected because they specifically targeted the problem of bacterial phase contrast segmentation [10, 12, 20]. Other bacteria-focused packages do exist, such as BactMAP, BacStalk, Cellprofiler, CellShape, ColiCoords, Cy-

tokit, MicroAnalyzer, MicrobeJ, Oufti, and Schnitzcells. However, these incorporate limited novel segmentation solutions and instead aim to provide tools for single-cell analysis such as lineage tracing and protein tracking [6,7,11,15–17,22,31–33] Furthermore, the segmentation that these programs perform depends broadly on thresholding and watershed techniques; therefore, Morphometrics is a reasonable proxy for their segmentation capabilities. We were unable to locate code or training data for BASCA at the time of writing [8]. Ilastik is a popular interactive machine-learning tool for bioimage segmentation, but training it using a manual interface was not feasible on a large and diverse dataset such as our own [18].

Among DNN approaches, Mask R-CNN was selected because it is a popular architecture for handling typical image segmentation tasks. It was also used in the segmentation and tracking package Usiigaci [21]. U-Net architectures have been implemented in a number of algorithms, including DeLTA, PlantSeg, MiSiC, StarDist, and Cellpose [9,12,14,19,23]. DeLTA was not included in this study because it operates similarly to MiSiC and was designed specifically for mother machine microfluidics analysis. DeLTA 2.0 was recently released to additionally segment confluent cell growth on agarose pads, but it remains quite similar to MiSiC in implementation [34]. PlantSeg could, in principle, be trained on bacterial micrographs, but we determined that its edge-focused design meant to segment bright plant cell wall features would not offer any advancements over the remaining U-Net methods that we tested.

2.2 Acquisition and annotation of ground-truth data

For training and benchmarking these algorithms, we acquired micrographs of assorted bacterial species representing diverse morphologies and optical characteristics. Many studies of bacteria involve mutations or treatments that cause extreme morphologies. To capture this additional diversity, we included wild-type and mutant bacteria grown in the presence of two beta-lactam antibiotics, cephalexin and aztreonam, and A22, which targets MreB [35]. Finally, based on general interest in microbial communities, we acquired images of mixtures of bacteria which display distinct morphologies and optical characteristics. In total, we

collected 4833 images constituting approximately 700,900 individual cells deriving from 14 species (Table 2.1). Next, we developed a streamlined approach for manual cell annotation and applied it to these images (Chapter 7), yielding 47,000 representative annotated cells that serve as our ground-truth dataset (`bact_phase`). We divided this data into a 27,500-cell training set and a 19,500-cell benchmarking set. Relevant cellular metrics (area, perimeter, mean diameter) did not differ substantially between the groups, confirming that the benchmarking set faithfully represents the training set (Fig. 2.1).

2.3 Training and tuning segmentation algorithms

All segmentation algorithms have tunable parameters to optimize performance on a given dataset. These include pre-processing such as image rescaling (often to put cells into a particular pixel diameter range), contrast adjustment, smoothing, and noise addition. Morphometrics and SuperSegger were manually tuned to give the best results on our benchmarking dataset. The neural network component of SuperSegger was not retrained on our data, as this is a heavily manual process involving toggling watershed lines on numerous segmentation examples. DNN-based algorithms are automatically trained using our dataset, and the scripts we used to do so are available in our GitHub repository. We adapted our data for MiSiC by transforming our instance labels into interior and boundary masks. Training documentation for MiSiC is not published. Training and evaluation parameters for MiSiC were tuned according to correspondence with the MiSiC authors. Cellpose and StarDist were trained with the default parameters provided in their documentation. StarDist has an additional tool to optimize image pre-processing parameters on our dataset, which we utilized.

To facilitate direct comparison of the algorithms, we first optimized their performance against our data. For the DNN approaches, each algorithm was trained on our dataset using developer-recommended parameters. Morphometrics and SuperSegger cannot be automat-

Table 2.1: Strains in BPCIS dataset

Species	Strain	Image count	Cell count	Cells in ground truth	Percent of ground truth	Notes
E. coli	DH5α	1378	98200	9719	20.5	Dense microcolonies grown on minimal media. Thin phenotype. ITPG-induced GFP cytosol marker. Time lapse. Imaged by the Wiggins lab.
		141	4536	4390	9.3	Dense microcolonies on LB. Time lapse. Imaged by the Wiggins lab.
		2	2277	-	-	Treatment with cephalexin. Tn7::GFP. Imaged by the Mougous lab.
	CS703-1 [36]	80	23169	1284	2.7	Mutant grown on LB and aztreonam. Elongated and branching phenotypes. Time lapse. Imaged by the Mougous lab.
S. flexneri	M90T	117	256618	1411	3	Treatment with A22. Tn7::GFP. Frames selected from time lapse after 1hr growth. Imaged by the Mougous lab.
		6	4482	4315	9.1	Treatment with cephalexin. Tn7::GFP. Frames selected from time lapse after 1hr growth. Imaged by the Mougous lab.
F. tularensis subsp *novicida*	U112	5	20166	496	1.1	Small and extremely low-contrast cells. Tn7::GFP. Imaged by the Mougous lab.
A. baylyi	ADP1 [37]	2169	60601	3332	7	Deletion of essential gene murA. Rounded phenotype. Time lapse. Imaged by the Wiggins lab.
		241	1313	1107	2.3	Deletion of essential gene ftsN. Filamentous phenotype. Time lapse. Imaged by the Wiggins lab.
		540	10013	2220	4.7	Deletion of essential gene dnaA. Filamentous phenotype. Time lapse. Imaged by the Wiggins lab.
B. thailandensis	E264 [38]	30	62005	5119	10.8	Selected panels from a self-intoxication experiment. Cells exhibit internal structure and low contrast in microcolonies. Tn7::GFP. Time lapse. Imaged by the Mougous lab.
H. pylori	LHS100 [39]	15	13014	-	-	Helical phenotype. Grown, fixed, and stained with Alexaflour 488 in the lab of Nina Salama. Imaged by the Mougous lab.
		19	1668	692	1.5	Treated with aztreonam. Filamentous, helical phenotype. Grown, fixed, and stained with Alexaflour 488 in the lab of Nina Salama. Imaged by the Mougous lab.
C. crescentus	NA1000 [40]	4	1787	753	1.6	Grown in HIGG media to induce stalk phenotype. Cultivation and imaging done in the lab of Yves Brun.
S. pristinaespiralis	NRRL 2958	17	2339	754	1.6	Grown on rich media to induce filamentous phenotype. Imaged by the Mougous lab.
V. cholerae	A1552 [41]	2	2627	2262	4.8	Cells have short but curved morphology and form dense, low-contrast microcolonies. Tn7::GFP. Obtained from the lab of Fitnat Yildiz. Imaged in the Mougous lab.
S. proteamaculans *E. coli*	568 DH5α	43	100146	1241	2.6	1:1 mixture. S.p. labelled via Tn7::GFP, *E. coli* unlabeled. Time lapse. Imaged in the Mougous lab.
P. aeruginosa *S. aureus*	PAO1 [42] USA300	3	2662	3684	7.8	1:1 mixture. *P. aeruginosa* labelled via Tn7::GFP, *S. aureus* unlabeled. Imaged in the Mougous lab.
P. aeruginosa *S. aureus* *V. cholerae* *B. subtilis*	PAO1 USA300 A1552 HM1350	21	33281	4664	9.8	1:1:1:1 mixture. *P. aeruginosa* and *V. cholerae* labelled via Tn7::GFP, *S. aureus* and *B. subtilis* labelled with red membrane dye. Imaged in the Mougous lab.
		4833	**700,904**	**47,443**	**100**	

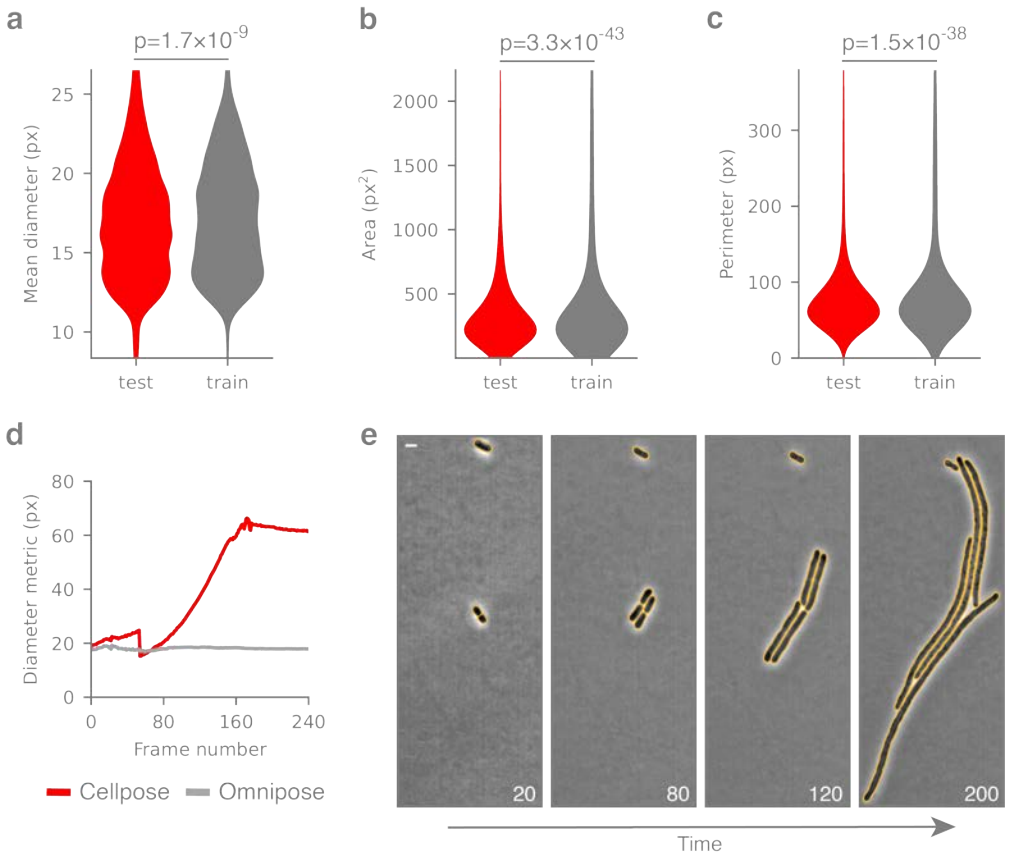

Figure 2.1: Size and morphology metrics are indistinguishable between cell populations included in training and test datasets. (a) Mean cell diameter, (b) cell area, and (c) cell perimeter calculated for our bacterial phase contrast ground truth dataset. P-values are displayed for the two-sided KS test. $n = 47,000$ (27,500 for training, 19,500 for testing). (d). Comparison of diameter metrics of a timelapse of elongated cell growth. The Cellpose diameter metric is the diameter of a circle with equivalent area. Omnipose diameter metric is proportional to the mean of the distance transform (Chapter 7). (e) Bacteria displayed are a single population analyzed of *A. baylyi* transformed with a $\Delta ftsN$ :: *kan* PCR fragment. Yellow lines indicate cell label boundaries. Scale bar, 1 µm.

ically optimized using ground-truth data; therefore, we manually identified settings that optimized the performance of these algorithms against our dataset (Chapter 7).

2.4 Quantifying segmentation performance

As a quantitative measure for algorithm performance, we compared their average Jaccard Index (JI) as a function of intersection over union (IoU) threshold (Fig. 2.2a) [43, 44]. IoU values lie between zero and one, with values greater than 0.8 marking the point at which masks become indistinguishable from ground truth by the expert human eye (Fig. 2.3) [43]. This analysis showed that DNN-based approaches significantly outperform other algorithms. However, substantial differences in performance within the DNN group were observed; Cellpose and StarDist outperform Mask R-CNN and MiSiC at high IoU thresholds. The performance of all algorithms varied greatly across images in the `bact_phase`dataset, with much of this variability delineated by cell type and morphology categories (Fig. 2.2b-g). Whereas all other algorithms exhibited visible segmentation errors in two of the three cell categories we defined, errors by Cellpose were only apparent in elongated cells (Fig. 2.2h-j).

All algorithms were evaluated on our benchmarking dataset with manually or automatically optimized parameters. We provide both the raw segmentation results for all test images by each tested algorithm as well as the models and model-training scripts required to reproduce our results. Before evaluating IoU or JI, small masks at image boundaries were removed for both the ground-truth and predicted masks. IoU and JI are calculated on a per-image basis and, where shown, are averaged with equal weighting over the image set or field of view.

Our new metric, the number of segmentation errors per cell, was calculated by first measuring the fraction of each predicted cell that overlaps with each ground truth cell. A predicted cell is assigned to a ground-truth cell if the overlap ratio is ≥ 0.75, meaning that at least three quarters of the predicted cell lies within the ground-truth cell. If several predicted

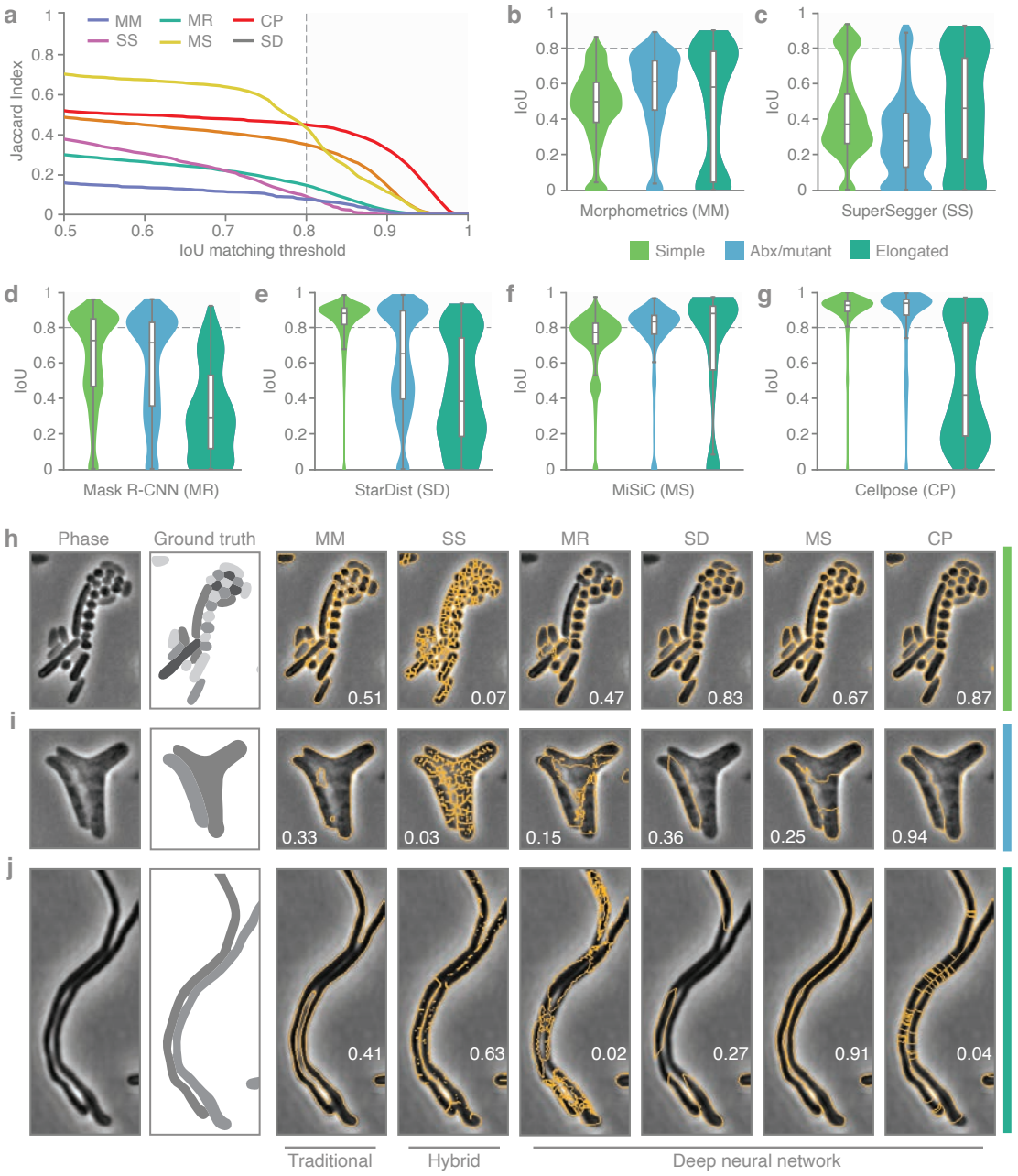

Figure 2.2: Quantitative comparison of segmentation methods distinguishes Cellpose as a high performing algorithm. (a-g) Comparison of segmentation algorithm performance on our `bact_phase` test dataset ($n = 19,538$ cells). (a) Overall performance measured by Jaccard Index (JI). The JI was calculated at the image level and values averaged across the dataset are displayed. (b-g) Algorithm performance partitioned by cell type (Simple, $n = 12,869$; Abx/mutant, $n = 6,138$; Elongated, $n = 531$). Images were sorted into types as defined in Supplemental Table 1 (Abx, antibiotic). Boxes centered on medians from Q1 to Q3, whiskers from Q1 $-$ 1.5IQR to Q3 $+$ 1.5IQR, IQR the inter-quartile range Q3 $-$ Q1. (h-j) Representative micrographs of cell type partitions analyzed in B-G, indicated by vertical bars at right. Ground-truth masks and predicted mask outlines generated by the indicated algorithm are displayed. Mean matched IoU values for cells shown are displayed within each micrograph. Bacteria displayed are (h) *V. cholerae*, *P. aeruginosa*, *B. subtilis*, *S. aureus*, (i) aztreonam-treated *E. coli* CS703-1, and (j) *S. pristinaespiralis*. All images scaled equivalently. Scale bar, 1 µm.

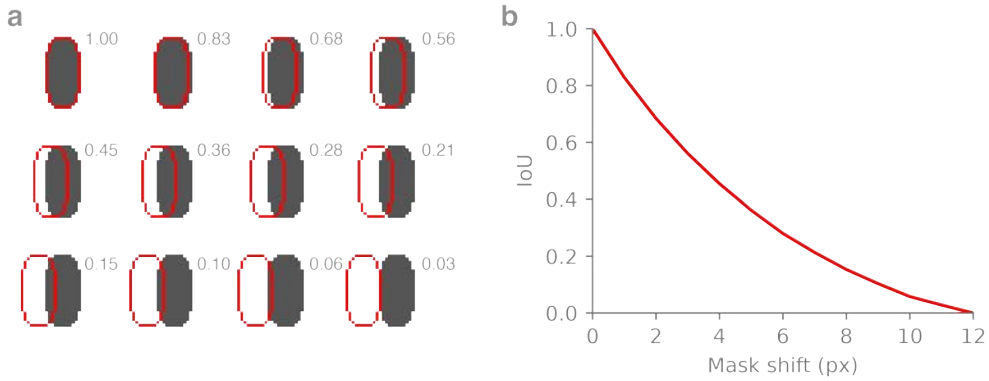

Figure 2.3: The relationship between IoU and segmentation accuracy. (a) Illustration of 0-12 pixel displacement of cell mask (red outline) and corresponding IoU values using a synthetic cell of typical bacterial size and resolution (solid black). (b) Quantification of the impact of mask shift on IoU values, determined using the synthetic cell shown in (a).

cells are matched to a ground-truth cell, the number of surplus matches is taken to be the number of segmentation errors. If no cells are matched to a ground-truth cell, then the error is taken to be 1.

2.5 Motivation for a new DNN-based segmentation algorithm

Our comparison revealed that Cellpose offers superior performance relative to the other segmentation algorithms we analyzed, and for this reason, we selected this algorithm for further development. Notably, even at the high performance levels of Cellpose, only 81% of predictions on our benchmarking dataset are above 0.8 IoU. This limits the feasibility of highly quantitative studies such as those involving subcellular protein localization or cell-cell interactions.

Cellpose utilizes two-step process. Its neural network first transforms an input image into several intermediate outputs, including a scalar probability field for identifying cell pixels in the next step (Fig. 2.4i-iii) [9]. Cellpose is unique among DNN algorithms by the addition of a vector field output (the flow field), which is defined by the normalized gradient of a heat

distribution from the median cell pixel coordinate (Fig. 2.4a). In the second step, this vector field directs pixels toward a global cell center via Euler integration, thereby segmenting cells based on the points at which pixels coalesce Fig. 2.4b). In contrast to other algorithms, this approach for reconstructing cell masks is size- and morphology-independent, insofar as the cell center can be correctly defined.

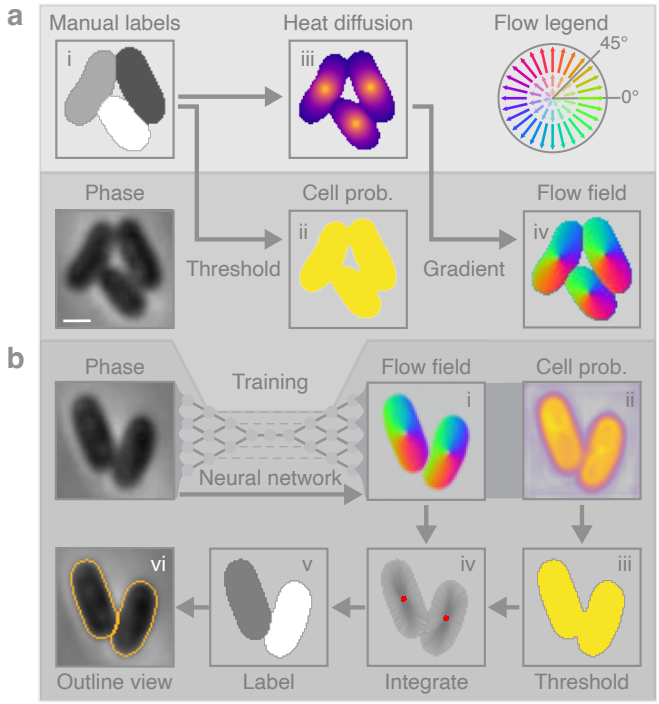

Figure 2.4: Details of the Cellpose algorithm. (a) Stages of the Cellpose training pipeline. Ground truth masks (i) are converted to cell probability (ii) by binary thresholding and a heat distribution (iii) by simulated diffusion from the median pixel coordinate. The flow field (iv) is defined by the normalized gradient of (iii). Color-magnitude representations of this vector field follow the flow legend diagram. The phase, cell probability, and flow fields are used to train the network. (b) Stages of the cellpose prediction pipeline. Phase images are processed by the trained cellpose network into the intermediate flow field and cell probability outputs (i-ii). A binary threshold is applied to the probability to identify cell pixels (iii). Pixels are Euler-integrated under the flow field until they converge at common points. Boundary pixel trajectories are depicted in iv. Each pixel is assigned a unique label corresponding to the center to which it converged (v). This segmentation result is commonly depicted in an outline view (vi). Bacteria shown are Escherichia coli. Scale bar 1 µm.

To understand the mechanisms behind Cellpose segmentation errors, we evaluated its performance as a function of cell size on our `bact_phase`dataset. We compared cell area against the number of segmentation errors, calculated as the number of redundant or missing masks corresponding to each ground-truth cell mask. This revealed a strong correlation between cell size and segmentation errors, with the top quartile of cells accounting for 83% of all errors (Fig. 2.5a). To understand the source of these errors, we inspected the flow field output of many poorly segmented cells across a variety of species and growth conditions. This showed that elongated cells, an important morphology often seen in both wild-type and mutant bacterial populations, are particularly susceptible to over-segmentation (Fig. 2.5b). We attribute this to the multiple sinks apparent in the corresponding flow fields. In the Cellpose mask reconstruction algorithm, pixels belonging to these cells are guided into multiple centers per cell, fragmenting the cell into many separate masks.

We hypothesized that the defect in Cellpose flow field output is a consequence of two distinct flow field types arising from our training dataset: those where the median pixel coordinate, or center, lies within the cell (97.8%) and those where it lies outside the cell (2.2%). In the latter, Cellpose projects the center point to the nearest boundary pixel, ultimately leading to points of negative divergence on cell peripheries that are chaotically distributed (Fig. 2.5c-e). On the contrary, non-projected centers maintain a uniform field magnitude along the entire boundary and adhere to the global symmetries of the cell (Fig. 2.6a,d). A similar issue is also encountered in cells with centers that fall close to but not outside of the boundary (Fig. 2.6b-d). Cells with a center point closer than 0.3 times the mean cell diameter (a factor of 0.2 off-center) to the boundary account for an additional 9.6% of our data. Neural networks can be exquisitely sensitive to the outliers in their training data [45]; therefore, we suspect that this small fraction of corrupt flow fields has significantly impacted the performance of Cellpose.

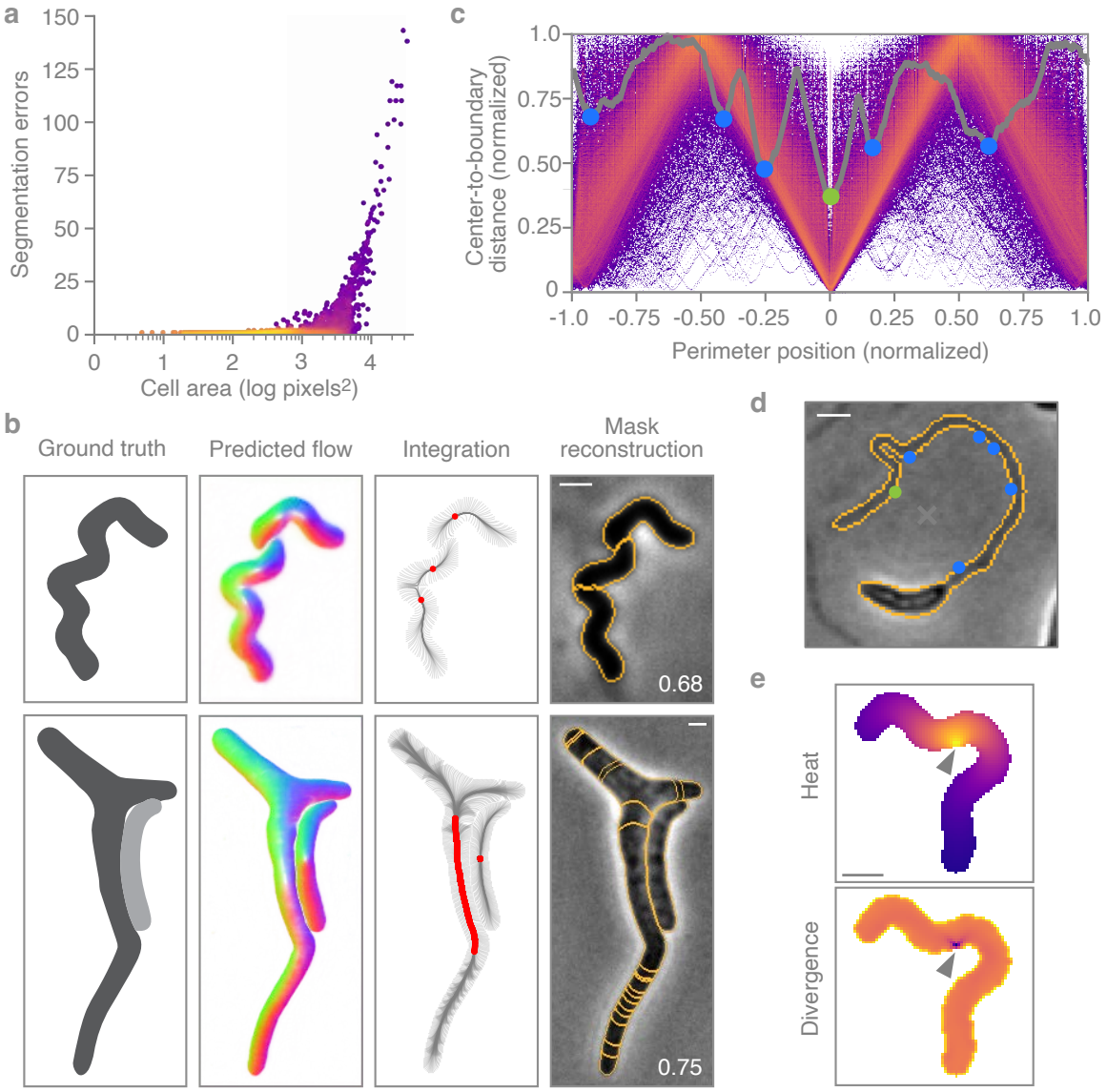

Figure 2.5: Cellpose over-segments extended, anisotropic cells. (a) Single-cell analysis of Cellpose segmentation error as a function of cell area. Color represents density on a log scale. Gray box represents the top quartile of cell areas (n=19,570). (b) Example images representative of 1,128 cells with segmentation errors in the top area quartile (n=4,887). Corresponding boundary pixel trajectories are shown in black and final pixel locations in red. Predicted mask overlays are shown with mean matched IoU values. Cellpose model `bact_phase` used in (a,b). (c) Analysis of stochastic center-to-boundary distances in our ground-truth dataset. Distance from the center (median pixel coordinate) to each boundary pixel is normalized to a maximum of 1. Position along the boundary is normalized from -1 to 1 and centered on the point closest to the median pixel. Center-to-boundary for the cell in (d) is highlighted in black. (d) Representative cell with median coordinate outside the cell body (black X). Cellpose projects this point to the global minima of this function (green dot), but several other local minima exist (blue dots). (e) The heat distribution resulting from a projected cell center (black arrow). The normalized gradient corresponds to the divergence shown. (d-e) represent n= 617 cells with projected centers in the training dataset. Bacteria displayed are (a,e) *H. pylori*, (b) *E. coli* CS703-1, both treated with aztreonam, and (d) *C. crescentus* grown in HIGG media. Scale bar is 1 µm.

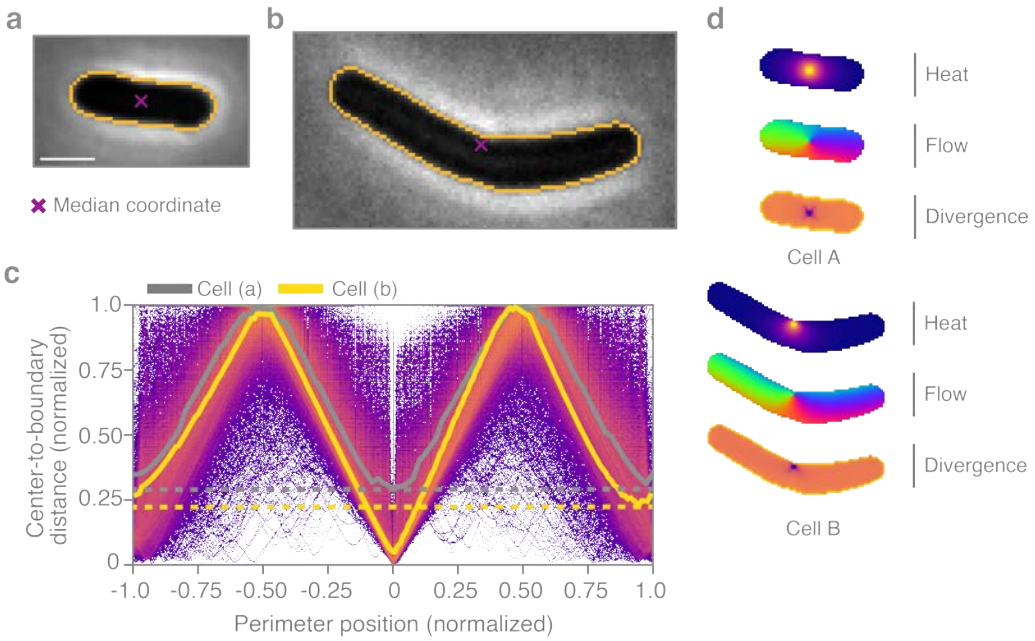

Figure 2.6: Median coordinates used to generate Cellpose ground-truth flow fields are asymmetrically localized for some bacterial cell morphologies. (a) Rod-shaped E. coli with symmetric median coordinate. Symmetry of the center is reflected in A by equal high and low points corresponding to the extremal points along the long and short axes of the cell. (b) Curved B. subtilis with median coordinate asymmetrically close to the cell boundary. This asymmetry is reflected in A by a secondary minimum above the global minimum corresponding to the diametrically opposing point along the short axis of the cell. (c) Center-to-boundary distance highlighted for cells A (black) and B (yellow) with non-projected median coordinates. Dashed lines indicate the larger of the two minima along the medial axis. (d) Flow fields generated by Cellpose for cells A and B. Scale bar is 1 µm. Images scaled equivalently.

Chapter 3

OMNIPOSE

Having thoroughly understood the faults of existing approaches, we next sought to develop a segmentation algorithm that operates independently of cell center identification (*i.e.*, without any assumption of round morphology). The algorithm we developed is based on the framework of Cellpose, which can be divided into five key components: file handling, neural network architecture, training objective functions, network predictions, and mask reconstruction. We made improvements to each of these components; however, the major innovations in our algorithm, which we named Omnipose, pertain to network predictions and mask reconstruction.

3.1 Prediction classes

Unlike the cell probability and center-seeking flow field upon which Cellpose is constructed, we built Omnipose on three distinct network outputs: a cell boundary probability map,[†] the distance field, and a flow field defined by the gradient of the distance field. The distance field (or distance transform) describes the distance at any point $\vec{x}$ in a bounded region Ω to the closest point on the boundary $\partial\Omega$. Notably, this widely utilized construct is one of the intermediate outputs of StarDist [23]. Whereas StarDist uses a distance field prediction to seed and assemble star-convex polygons, Omnipose implements the distance field as a replacement to the cell probability output of Cellpose.

The use of a distance field has several advantages. First, the distance field is more structured than a binary probability map and offers higher fidelity thresholding to seed cell masks.

[†]This output is optional, and has since been retired in more recent versions of Omnipose. It was useful only as a means to improve predictions of distance and flow at cell boundaries. Our reasoning was that the network is forced to "focus" on cell boundaries with this output. Later additions to loss functions have achieved the same effect.

Second, the distance field is defined by the Eikonal equation $|\vec{\nabla}\Phi(\vec{x})| = 1$, and so its gradient – the flow field of Omnipose – has unit magnitude throughout the bounded region for which it is calculated. This leads to faster convergence and better numerical stability when compared to alternative solutions producing similar fields (*e.g.*, screened Poisson; Fig. 3.1a, Chapter 7). Third, the distance field is independent of morphology and topology, meaning that it is applicable to all cell shapes and sizes. Lastly, the resulting flow field points uniformly from cell boundaries toward the local cell center, coinciding with the medial axis (skeleton) that is defined by the stationary points of the distance field (Fig. 3.1b). This critical feature allows pixels to remain spatially clustered after Euler integration, solving the problem of over-segmentation seen in Cellpose.

One challenge to implementing a distance-field-based approach is that traditional distance field algorithms like FMM (Fast Marching Method) are sensitive to boundary pixilation [46], causing artifacts in the flow field that extend deep into the cell. These artifacts are sensitive to pixel-scale changes at the cell perimeter, which we reasoned would interfere with the training process. To solve this problem, we developed an alternative approach based on FIM (Fast Iterative Method) that produces smooth distance fields for arbitrary cell shapes and sizes (Fig. 3.1, Fig. 3.2a, and Chapter 7) [47]. The corresponding flow field is relatively insensitive to boundary features at points removed from the cell boundary, a critical property for robust and generalized prediction by the Cellpose network.

The use of the distance field additionally required a unique solution for mask reconstruction. Whereas the pixels in a center-seeking field converge on a point, standard Euler integration under our distance-derived field tends to cluster pixels into multiple thin fragments along the skeleton, causing over-segmentation (Fig. 3.2b). We solved this with a suppression factor of $(t+1)^{-1}$ in each time step of the Euler integration (Fig. 3.2c). This reduces the movement of each pixel after the first step $t = 0$, facilitating initial cell separation while preventing pixels from clustering into a fragmented skeleton formation. The wider point distribution resulting from our suppression factor allows pixels to remain connected, thereby generating a

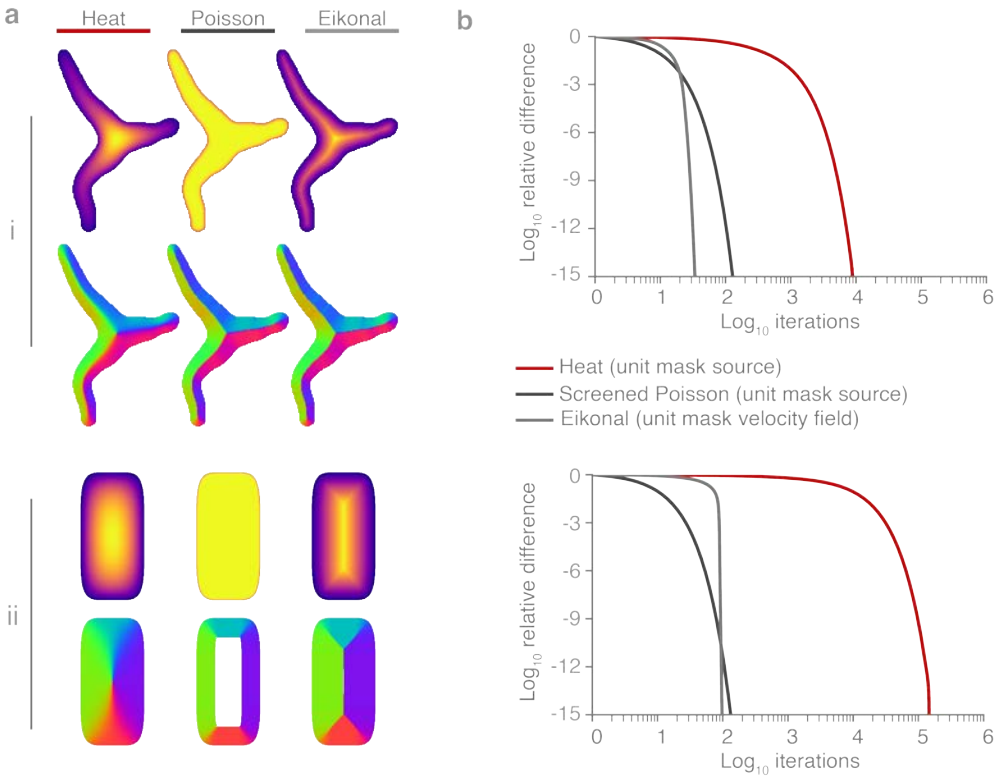

Figure 3.1: The Eikonal equation provides a fast and accurate flow field calculation for diverse cell morphologies and sizes. (a) Partial differential equation solutions (top rows) and corresponding flow fields (bottoms rows) calculated for two examples cells (i, ii) using a relaxation algorithm for the heat, Poisson and Eikonal equations. Cell (i) is drawn from our dataset (mean diameter 37 px) and cell (ii) is a synthetic rod-shaped cell (mean diameter 192 px). (b) Convergence measured by the average difference at each iteration (maximum normalized to 1) for cells (i,ii).

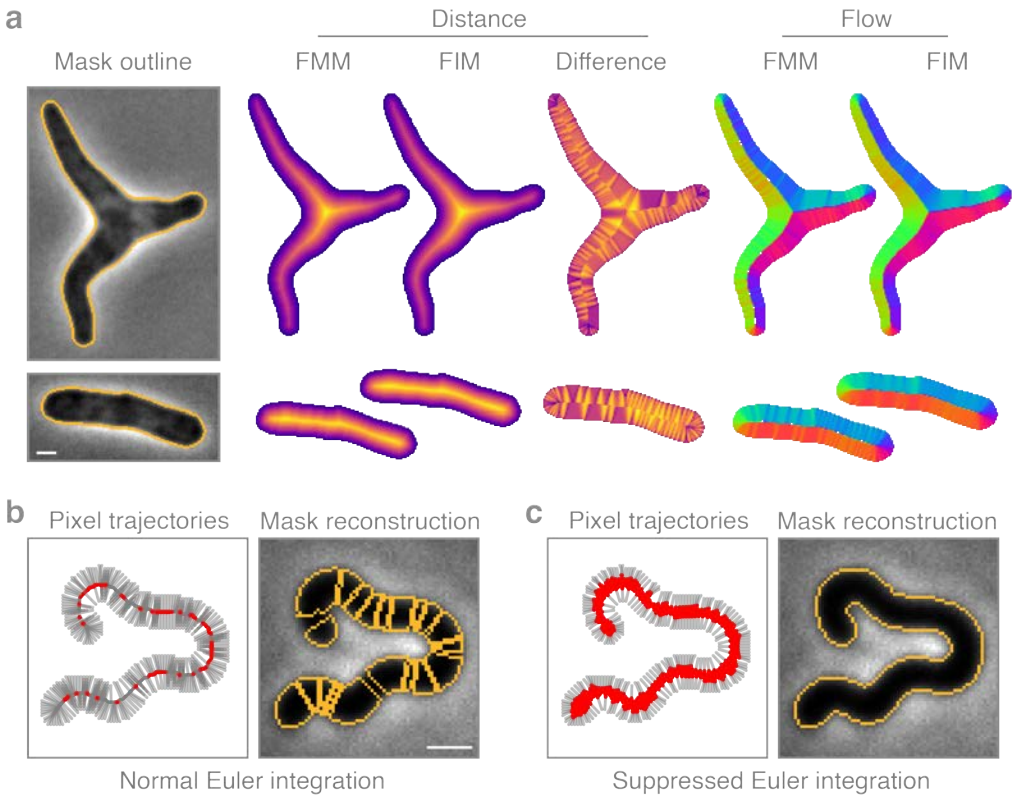

Figure 3.2: Core innovations of Omnipose. (a) Comparison of distance field algorithms and corresponding flow fields on ground truth masks. Fast Marching Method (FMM) produces ridges in the distance field resulting from pixelation on the cell mask boundary. Our smooth FIM algorithm minimizes these features. The difference image (FIM FMM) highlights artifacts in the FMM method. Flow fields are calculated as the normalized gradient of the distance field. Boundary pixelation affects the FMM flow field deep into the cell, regardless of cell size. (b-c) Comparison of mask reconstruction algorithms on a smooth flow field. (b): boundary pixel trajectories and resulting mask outlines from standard Euler integration. (c): Trajectories and mask outlines under suppressed Euler integration. Red dots indicate the final positions of all cell pixels, not only the boundary pixels for which trajectories are displayed. Bacteria displayed are (a) *E. coli* CS703-1 and (b, c) and *H. pylori*, both treated with aztreonam. Scale bars, 1 µm. Images are representative of 1,299 *E. coli* and 701 H. pylori cells in the total ground truth dataset, respectively.

single mask for each cell in conjunction with a standard automated pixel clustering algorithm (*e.g.*, DBSCAN) [48].

3.2 Loss functions

Loss functions define the energy landscape in which the network parameters are optimized. Two loss function constructions are used in Cellpose and Omnipose. First is mean squared error (MSE), defined as

$$\mathrm{MSE}(p, \hat{p}) = \frac{1}{n} \sum_{i=1}^{n} (p_i - \hat{p}_i)^2$$

where n is the number of samples (pixels), p_i are predictions, and $\hat{p}_i$ are ground truth. MSE is commonly used in regression problems. Second is binary cross entropy with logits (BCEL), defined as

$$\mathrm{BCE}_L(p, \hat{p}) = -\frac{1}{n} \sum_{i=1}^{n} p_i \cdot \log(\sigma(\hat{p}_i)) + (1 - p_i) \cdot \log(1 - \sigma(\hat{p}_i))$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. BCE is commonly used for binary classification tasks, and the logits variant is especially useful in this context because it symmetrizes the $(0, 1)$ binary input to $(-5, 5)$. This corresponds to the symmetric range of the flow field components $(-1, 1)$, which we multiply by 5 so that all outputs are in the range $(-5, 5)$.

Whereas Cellpose uses MSE on the flow field components and BCE_L on the cell probability field, Omnipose uses BCE_L on the boundary field, MSE on the distance field and flow field components, and then adds multiple additional MSE terms on the gradient of the distance field as well as the divergence and norm of the flow field. These changes tot he energy landscape were observed to improve convergence time and prediction quality. Many later improvements and additions/substitutions were made to the Omnipose loss function (see Section 4.5).

3.3 Omnipose demonstrates unprecedented segmentation accuracy

To benchmark the performance of Omnipose, we trained a model (`bact_phase_omni`) based on our bacterial phase contrast dataset. Remarkably, across the IoU threshold range 0.5–1, the accuracy of Omnipose significantly exceeds that of Cellpose using a corresponding model (`bact_phase_cp`) (Fig. 3.3a). This difference in performance between the algorithms is particularly pronounced within the high IoU range (0.8–1.0). Bacteria are 0.5–5 µm in scale and are typically imaged with a calibrated pixel size of about 0.1 µm, resulting in cells and cell labels that are 5–50 px across [49]. Quantitative measurements at this scale require pixel-level accuracy, corresponding to IoU values above 0.8 (Fig. 2.3). Thus, Omnipose is uniquely suited for the microscopic analysis of bacterial cells.

To dissect the contributions of the individual Omnipose innovations to the overall performance of the algorithm, we isolated the mask reconstruction component of Omnipose and applied it to the Cellpose network output. This augmentation of Cellpose modestly improved its performance across all IoU thresholds (Fig. 3.3a). Based on this, we attribute the remaining gains in performance by Omnipose to its unique network outputs (boundary, distance, and flow) and to our improvements to the Cellpose training framework. The latter includes numerous custom loss functions, the use of an alternative optimizer (RAdam), and image augmentations (Chapter 7).

Our analyses illuminated critical flaws in prior DNN-based approaches for the segmentation of elongated cells, effectively preventing these algorithms from generalizable application to bacteria (Fig. 2.2). To determine whether Omnipose overcomes this limitation, we evaluated its performance as a function of cell area. Cell area serves as a convenient proxy for cell length in our dataset, which is composed of both branched and unbranched elongated cells. Whereas the Cellpose cell error rate remains above 15% across all cells, the Omnipose error rate does not exceed 5% before the 90th percentile of cell area is reached (Fig. 3.3b). Thus,

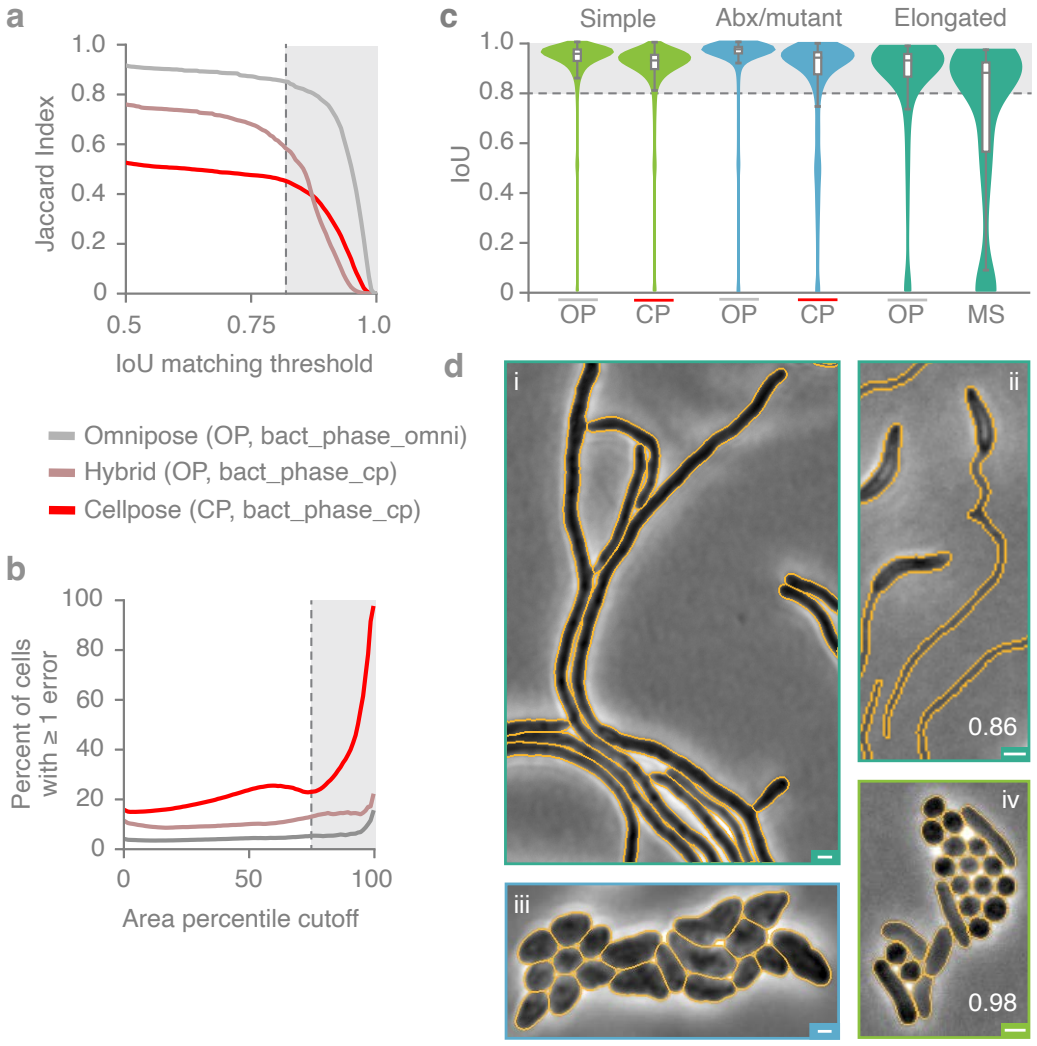

Figure 3.3: Omnipose substantially outperforms Cellpose on elongated cells. (a) Overall performance of Omnipose (`bact_phase_omni`) and Cellpose (`bact_phase_cp`) measured by Jaccard Index (JI). The hybrid method (gray) uses the original center-seeking flow output of `bact_phase_cp` and the mask reconstruction of Omnipose. Gray box represents IoU ≥ 0.8. $n = 19,570$ cells in the test set. (b) Quantification of segmentation performance by cell size. The percent of cells with at least one segmentation error is computed for cells in each area percentile group from 1 to 100. Gray box represents the top quartile. (c) Omnipose IoU distribution on our dataset compared to the next highest performing algorithm in each of three cell categories (Simple, $n = 12,869$; Abx/mutant, $n = 6,138$; Elongated, $n = 531$). Boxes centered on medians from Q1 to Q3, whiskers from Q1 $-$ 1.5IQR to Q3 + 1.5IQR, IQR the inter-quartile range Q3 $-$ Q1. (d) Example micrographs and Omnipose segmentation. Mean matched IoU values shown. Bacteria displayed are (i) *S. pristinaespiralis*, (ii) *C. crescentus* grown in HIGG media, (iii) *S. flexneri* treated with A22, (iv) mix of *P. aeruginosa*, *S. aureus*, *V. cholerae*, and *B. subtilis*. Scale bars, 1 µm.

Omnipose performance is independent of cell size and shape, including those cells with complex, extended morphologies (Fig. 3.3c,d, Fig. 3.4a).

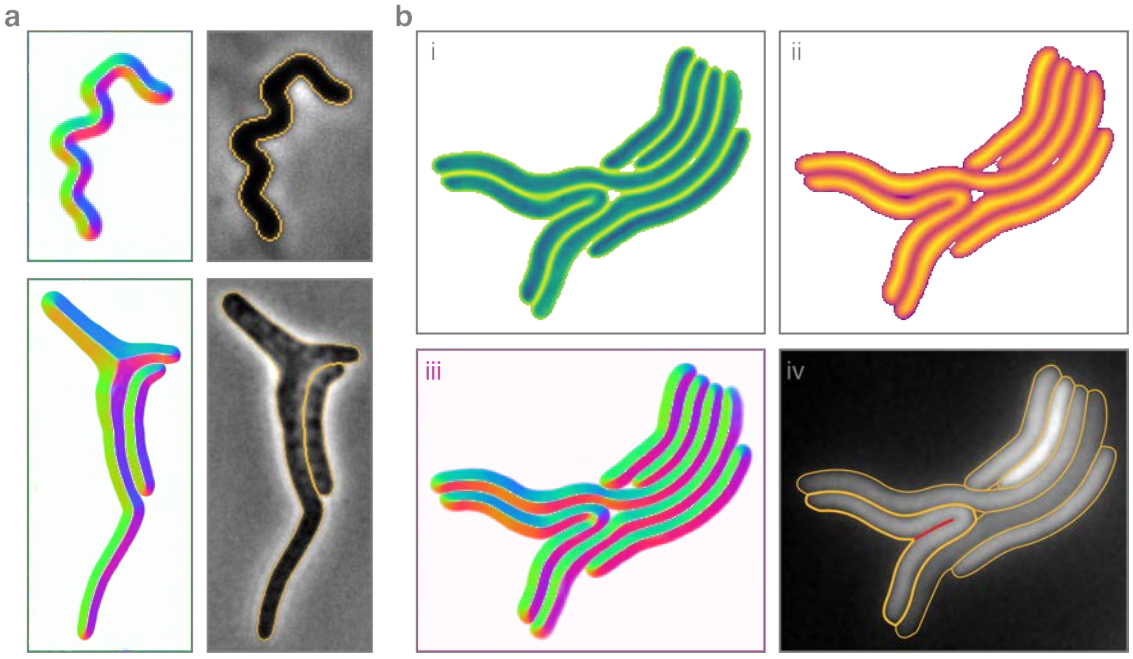

Figure 3.4: Omnipose output on diverse cell morphologies. (a) Omnipose flow and segmentation corresponding to the cells of Fig. 2.5b. (b) (i-iii) Boundary, distance, and flow output using `bact_fluor_omni` model on *S. flexneri* treated with $10\,\mu g\,mL^{-1}$ cephalexin. (iv) Overlaid mask outlines. The cell in bold yellow is missing the self-contact boundary in red.

3.4 Nematode segmentation

We have shown that the features of Omnipose improve phase-contrast bacterial segmentation performance beyond that of Cellpose. Like other DNN-based algorithms, Omnipose can also segment images acquired using different modalities and composed of alternative subjects when trained on a representative dataset. To evaluate Omnipose on images acquired using a modality distinct from phase contrast, we curated a dataset containing cytosol and membrane fluorescence in 33,200 bacterial cells (`bact_fluor`). In brief, this was achieved by

applying the labels of cells with fluorescence signal in our phase contrast ground-truth images to their corresponding fluorescence channels (Chapter 7). As expected, we found that the enhanced performance of Omnipose (`bact_fluor_omni`) relative to Cellpose (`bact_fluor_cp`) on morphologically diverse cells translates to fluorescence images (Fig. 3.5a-d, Fig. 3.4b).

We next sought to investigate the potential utility of Omnipose in the segmentation of non-bacterial subjects. The nematode *Caenorhabditis elegans* is a widely studied model organism with an overall morphology similar to elongated bacteria [50]. At just one millimeter in length, *C. elegans* phenotypes are often analyzed by time-lapse microscopy; therefore, there is significant interest in segmentation methods that enable accurate tracking [51]. We obtained, annotated, and trained Omnipose on two publicly available, low-resolution microscopy datasets composed of *C. elegans* images: time-lapse frames from the Open Worm Movement database [52] and frames containing fields of assorted live or dead *C. elegans* from the BBBC010 dataset [53]. Many images in this combined worm dataset contain debris and are of heterogenous quality, yet 83% of masks predicted by Omnipose match or exceed the 0.8 IoU threshold (Fig. 3.5f). At the low resolution of our combined *C. elegans* dataset, the worms approximate bacteria in shape and diameter (in pixels). Indeed, we found that a generalist model trained on both our extensive `bact_phase`dataset and our worm dataset (`worm_bact_omni`) exhibited equivalent or higher performance relative to the specialist worm model (`worm_omni`). We also trained an Omnipose model using a custom collection of high-resolution *C. elegans* images (`worm_high_res`) and found that the algorithm can successfully segment these images despite the complex internal structure (Fig. 3.5h).

3.5 Benchmarking beyond bioimages

The `cyto2` dataset is a large collection of images and corresponding ground-truth annotations submitted by users that expands upon the original `cyto` dataset developed to evaluate Cellpose [9, 43]. This dataset includes many non-cell images (*e.g.*, rocks, onions) as well as multi-channel fluorescence images (*e.g.*, cytosol and nuclear stains in mammalian cells)

(Fig. 3.5i, Fig. 3.6). We found that Omnipose offers a modest improvement in performance relative to Cellpose on the `cyto2` dataset (Fig. 3.5e), and this was achieved without compromising the segmentation rate ($\sim$ 1 image per second).

3.6 3D segmentation

Two-dimensional imaging allows the characterization of cells within constrained environments, yet many phenomena of interest can only be studied in natural, three-dimensional contexts. We modified each explicitly 2D component of the network architecture, outputs, and mask reconstruction elements of Omnipose to enable direct segmentation of 3D (or higher order) data. This contrasts with the solution adopted for Cellpose (Cellpose3D), which approximates a 3D field by combining the 2D flow components predicted on orthogonal 2D volume slices [9].

We compared Cellpose3D to Omnipose on a publicly available *A. thaliana* lateral root primordia dataset acquired using confocal microscopy [19]. An Omnipose model was trained on 6 volumes representing 931 cells in total (`plant_omni`), whereas Cellpose was trained on 3,070 slices of these volumes (`plant_cp`) (Chapter 7). Consistent with our results in 2D, Omnipose provided more accurate segmentation results than Cellpose, particularly amongst elongated cells in the dataset (Fig. 3.7). We note that the absolute scores for both algorithms are low and attribute this in part to inaccurate ground-truth masks (Fig. 3.8a,b). Indeed, in certain instances, Omnipose-predicted masks appear to be more accurate than the ground truth (Fig. 3.7b, Fig. 3.8b). Interestingly, the 3D field reconstruction made in Cellpose from 2D `plant_cp` slice predictions recapitulates much of the direct 3D field predictions of `plant_omni` (Fig. 3.8c). We reason that this is because the local 2D cell slice centers of the Cellpose flow field can be coincident with the global 3D cell skeletons of the Omnipose flow field. Taken together with our results on diverse cell types and fluorescence images, we conclude that Omnipose maintains the multi-modal segmentation capabilities of Cellpose

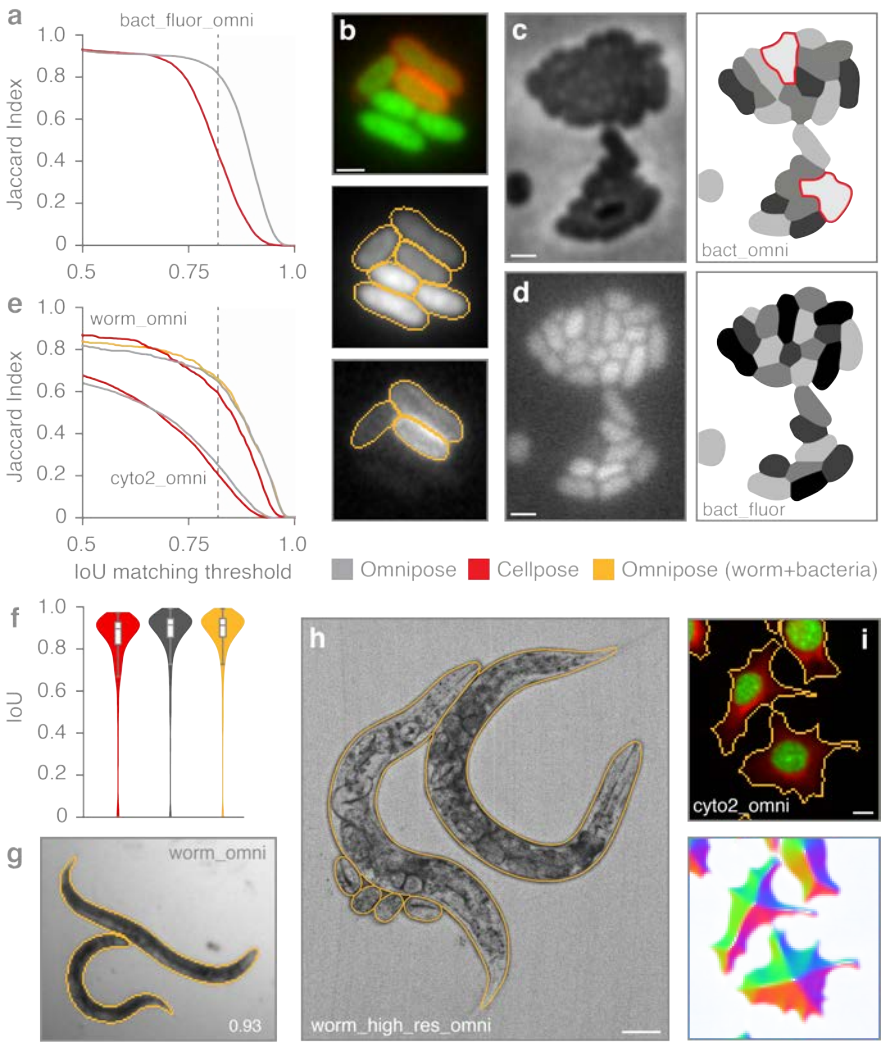

Figure 3.5: Omnipose models trained and evaluated on assorted imaging modalities and subjects. Relevant Omnipose model names provided (also see Table 3.1). (a) Performance of bacterial fluorescence Omnipose (`bact_fluor_omni`) and Cellpose (`bact_fluor_cp`) models. Test set consists of $n = 14,587$ cells. (b) Fluorescence micrograph (top) and corresponding `bact_fluor_omni` segmentation results of *B. thailandensis* expressing cytoplasmic GFP (segmentation, middle) or outer-membrane localized mCherry (segmentation, bottom). (c,d) *F. tularensis* subsp *novicida* segmentation using phase contrast (`bact_phase_omni`) (c) or `bact_fluor_omni` (d) Omnipose models. Two significant `bact_phase_omni` segmentation errors are highlighted in red. Scale bars, 1 µm. (e) Performance of Omnipose models trained on cyto2 (`cyto2_omni`, $n = 10,232$) and *C. elegans* (`worm_omni`) datasets versus corresponding Cellpose models. Results for Omnipose and Cellpose trained on either *C. elegans* alone (gray, red) or Omnipose on *C. elegans* and bacterial data (yellow) are shown. `cyto2` test dataset: $n = 10,232$ cells. *C. elegans* test dataset: $n = 1264$ worms. (f) IoU distribution for the masks predicted by each method on our *C. elegans* test dataset ($n = 1,264$). Boxes centered on medians from Q1 to Q3, whiskers from Q1 − 1.5IQR to Q3 + 1.5IQR, IQR the inter-quartile range Q3 − Q1. (g) Example segmentation of *C. elegans* in the BBBC010 dataset used in (e,f). IoU score shown. Scale bar unavailable. (h) Example segmentation of high-resolution *C. elegans* (minimum projection brightfield). Scale bar is 50 µm. (i) Example multi-channel Omnipose segmentation in the `cyto2` dataset. Scale bar, 3 µm.

Table 3.1: Models and datasets

Model name	**Dataset**	**Training**

`cyto2_omni`	`cyto2` dataset [9]	`--n_epochs` 4000
		`--RAdam`
		`--diameter` 30
		`--omni`

`bact_phase_omni`	`bact_phase` (BPCIS, Bacterial Phase Contrast for Image Segmentation)	`--n_epochs` 4000
		`--RAdam`
		`--diameter` 0
		`--omni`

`bact_phase_cp`		`--n_epochs` 500 `--diameter` 0

`bact_fluor_omni`	`bact_fluor`, bacterial fluorescence (cytosol and membrane)	`--n_epochs` 4000
		`--RAdam`
		`--diameter` 0
		`--omni`

`bact_fluor_cp`		`--n_epochs` 500 `--diameter` 0

`worm_omni`	`worm`, *C. elegans* brightfield	`--n_epochs` 4000
		`--RAdam`
		`--diameter` 0
		`--omni`

`worm_bact_omni`	`worm` and `bact_phase`	
`worm_cp`	`worm`	`--n_epochs` 500 `--diameter` 0
`worm_high_res`	*C. elegans* brightfield minimum projections dataset from Luca Rappez.	`--n_epochs` 4000 –RAdam `--diameter` 60 –omni

`plant_omni`	*A. thaliana* root cell primordia. Adapted from Wolny et al. [19] and downscaled by 1/3.	`--n_epochs` 4000
		`--RAdam`
		`--diameter` 0
		`--omni`

`plant_cp`		`--n_epochs` 500 `--diameter` 0

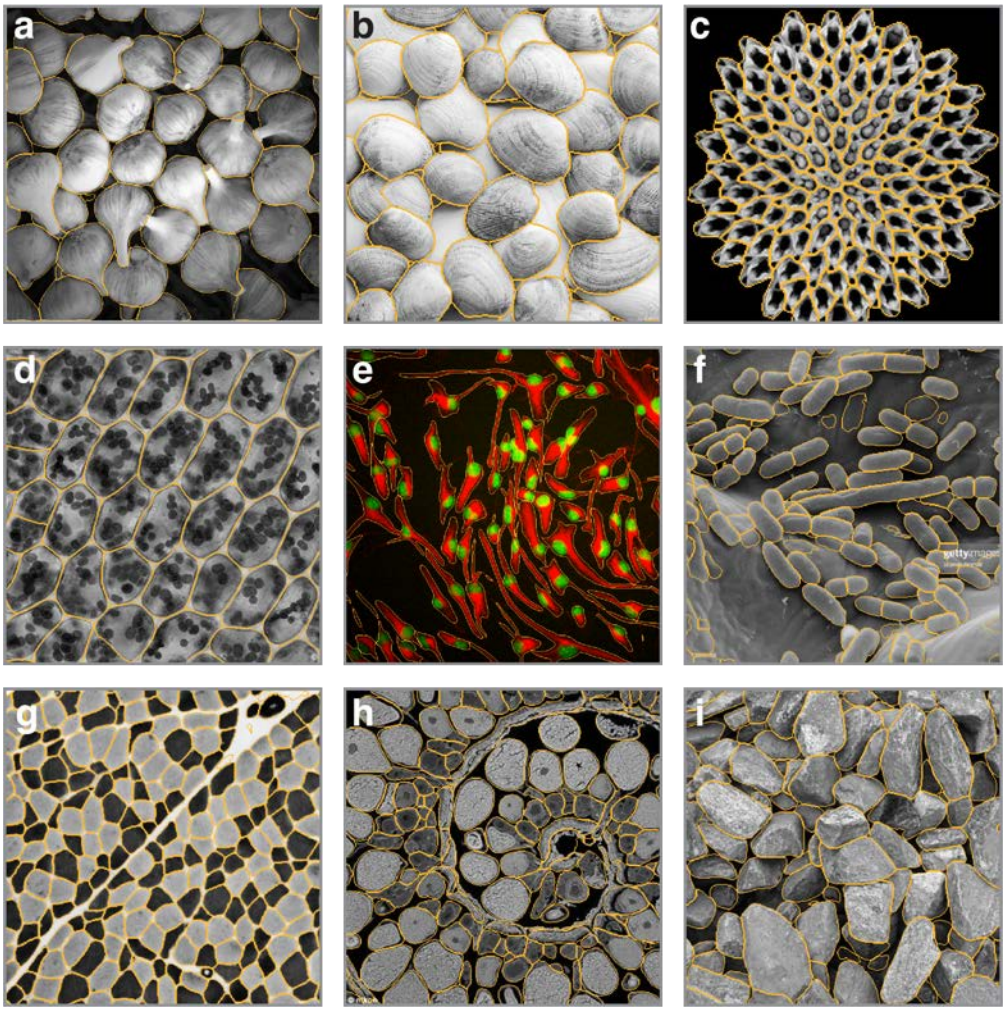

Figure 3.6: Omnipose accurately segments diverse images from the `cyto2` test dataset. (a-i) Selection of images from the `cyto2` test dataset with superimposed outlines representing Omnipose `cyto2_omni` model segmentation. Subjects are not defined, as the `cyto2` dataset lacks metadata and is comprised of anonymized contributions.

while adding the ability to perform accurate segmentation of a substantially broader range of cellular morphologies present within 2D and 3D data.

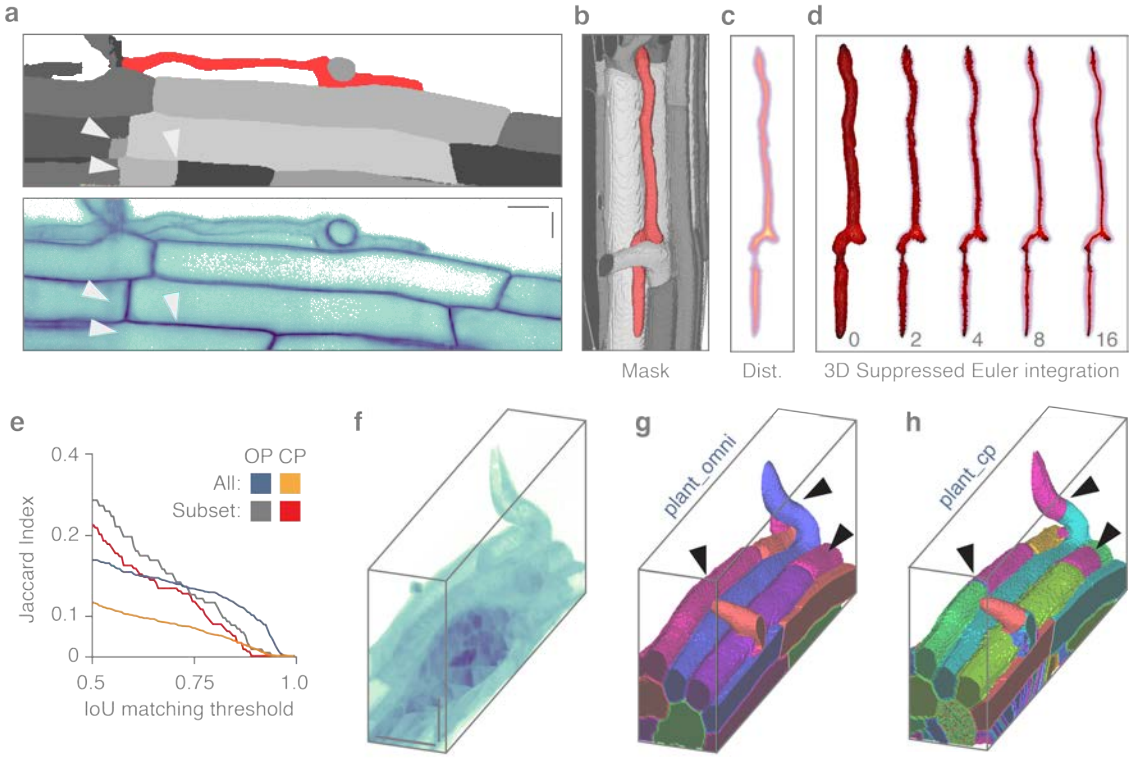

Figure 3.7: Omnipose can be applied to 3D datasets. (a) Volume slice in the *A. thaliana* 3D training set. Black arrows denote ground truth over-segmentation. Red corresponds to cell in panels b-d. (b) Selected cell in context of ground truth data. (c) Ground-truth distance field for selected cell (b). (d) Steps of the suppressed Euler integration for selected cell (b). Boundary pixels shown, colored by overall displacement (dark to light red). (e) Performance of Omnipose (OP, `plant_omni`) and Cellpose (CP, `plant_cp`) on the full test dataset ($n = 604$) and on a subset without excluded regions (Chapter 7) that is represented in panels f-h ($n = 73$). (f-h) Ground-truth masks (f) and segmentation results of *A. thaliana* using Omnipose (g) and Cellpose (h). Scale bars, 20 20 µm.

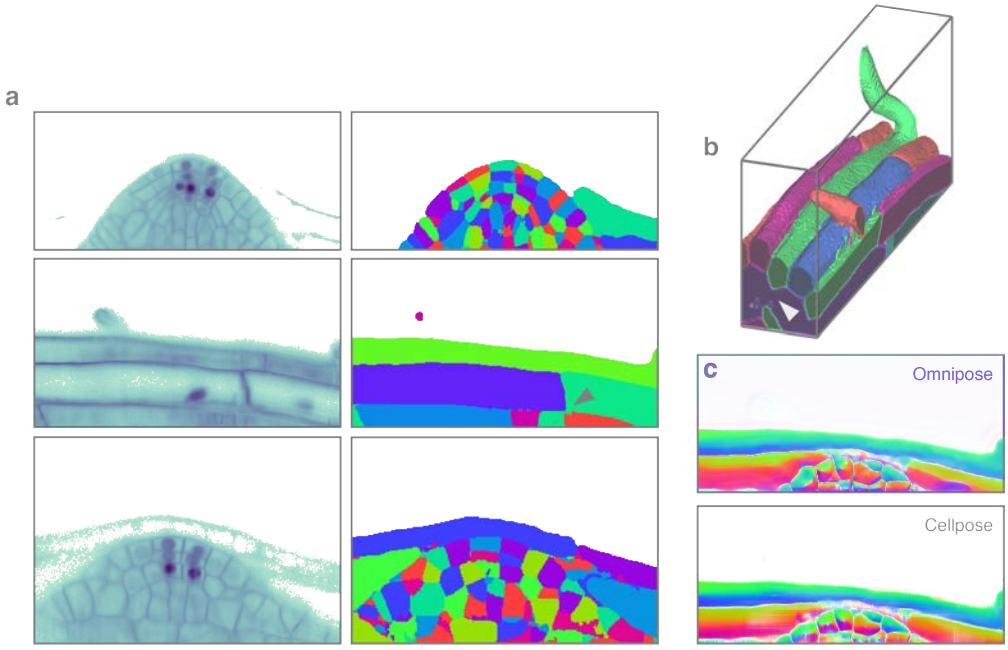

Figure 3.8: Errors in the three-dimensional *A. thaliana* ground-truth dataset impact training and performance metrics of Ominpose and Cellpose. (a) Slices of ground truth training set volumes. Images show errant pixels and apparent flaws in ground truth masks. Black arrow indicates one instance of a joined mask. (b) Ground truth masks for the test volume in Fig. 3.7e-h. Two cells are joined under one label (black arrow). (c) Slices of the predicted Omnipose and Cellpose flow fields through the middle of the volume in (b).

3.7 Sensitive detection of cell intoxication

Our laboratory recently described an interbacterial type VI secretion system-delivered toxin produced by *Serratia proteamaculans*, Tre1 [54]. We showed that this toxin acts by ADP-ribosylating the essential cell division factor FtsZ; however, we were unable to robustly evaluate the consequences of Tre1 intoxication on target cell morphology owing to segmentation challenges. Here we asked whether Omnipose is able to detect cellular intoxication by Tre1. To this end, we incubated *S. proteamaculans* wild-type or a control strain expressing inactive Tre1 ($tre1^{E415Q}$) with target *E. coli* cells and imaged these mixtures after 20 hours. Owing to the improved segmentation accuracy of the Omnipose `bact_phase` model, we were able to include dense fields of view and incorporate $\sim 300,000$ cells in our analysis.

Among the cells identified by Omnipose, we found a small proportion were elongated and much larger than typical bacteria (Fig. 3.9a,b and Fig. 3.10a). These cells were only detected in mixtures containing active Tre1, and the apparent failure of the cells to septate is consistent with the known FtsZ-inhibitory activity of the toxin. The *S. proteamaculans* strain background we employed in this work expresses the green fluorescent protein. Corresponding fluorescence images allowed us to unambiguously assign the enlarged cell population to *E. coli* (Fig. 3.9c). Next, we subjected the same images to cell segmentation with StarDist, Cellpose, and MiSiC, the three top-performing algorithms in our initial survey. Each of these algorithms fail to identify this population of cells to high precision (Fig. 3.9d,e). Close inspection reveals three distinct modes of failure (Fig. 3.9e and Fig. 3.10a,b). In the case of StarDist, elongated (non-star-convex) cells are split into multiple star-convex subsets that do not span the entire cell. Cellpose detects entire elongated cells but fragments them into a multitude of smaller masks. Conversely, MiSiC detects all cells but fails to properly separate them, thereby exaggerating the area measurement in many cases, including *E. coli* cells in the control experiment. These data illustrate how the enhanced cell segmentation performance of Omnipose can facilitate unique insights into microbiological systems.

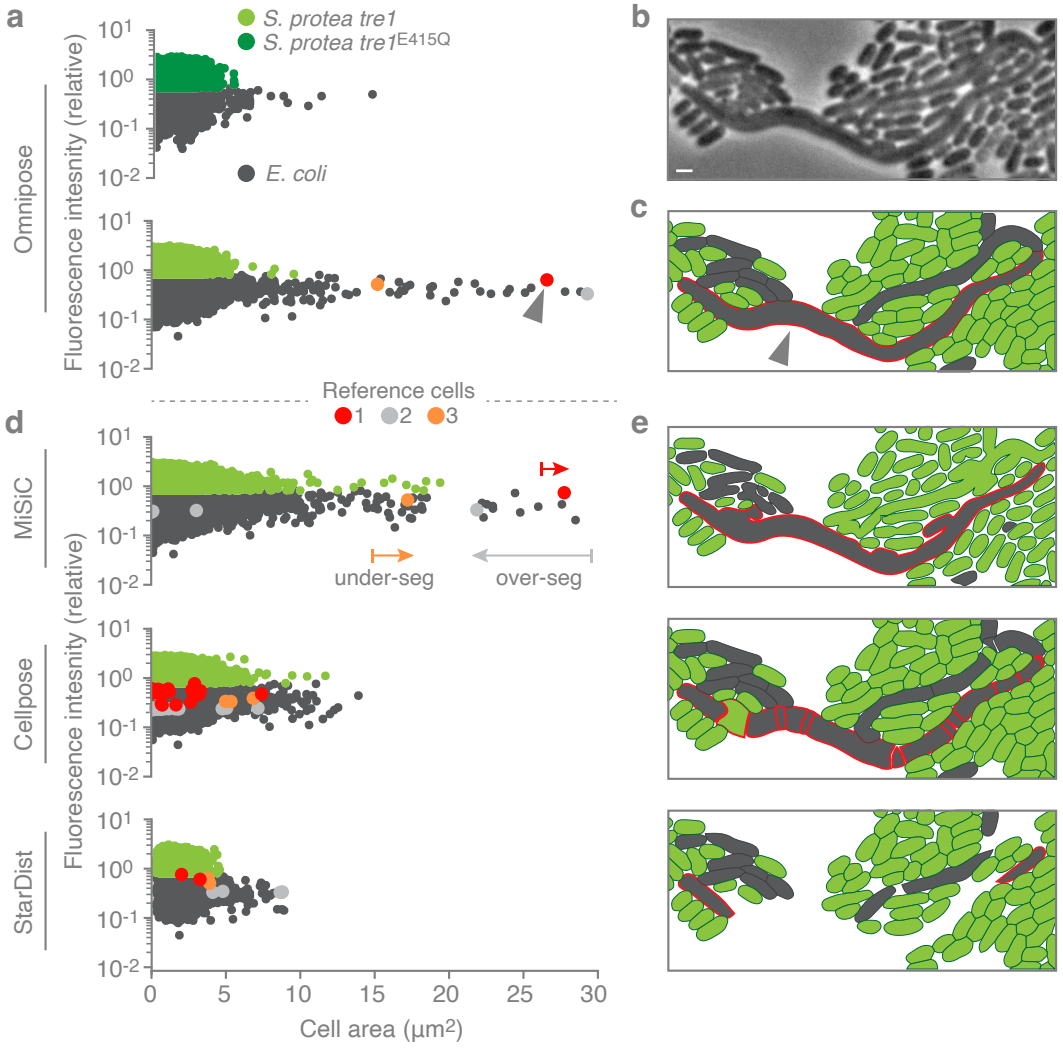

Figure 3.9: Omnipose facilitates the accurate identification of intoxicated *E. coli*. (a) Fluorescence/area population profile according to Omnipose segmentation (`bact_phase_omni`) in control and experimental conditions. K-means clustering on GFP fluorescence distinguishes *S. proteamaculans tre1*/*tre1*^{E415Q} (light/dark green markers) from *E. coli* (gray markers). $n = 209,000$ cells in experimental population (*S. proteamaculans tre1*) and $n = 85,260$ cells in control group (*S. proteamaculans tre1*^{E415Q}). (b) Example of extreme filamentation of *S. proteamaculans* in response to active Tre1. (c) Omnipose accurately segments all cells in the image. Largest cell indicated with black arrow. (d) MiSiC predicts large cell masks over both species. Cellpose (`bact_phase_cp`) and StarDist fail to predict any cells above 15 µm^2. (e) Example segmentation results highlighting typical errors encountered with MiSiC (under-segmentation), Cellpose (over-segmentation), and StarDist (incomplete masks). Mask mergers cause some *E. coli* to be misclassified as *S. proteamaculans*. Scale bar is 1 µm.

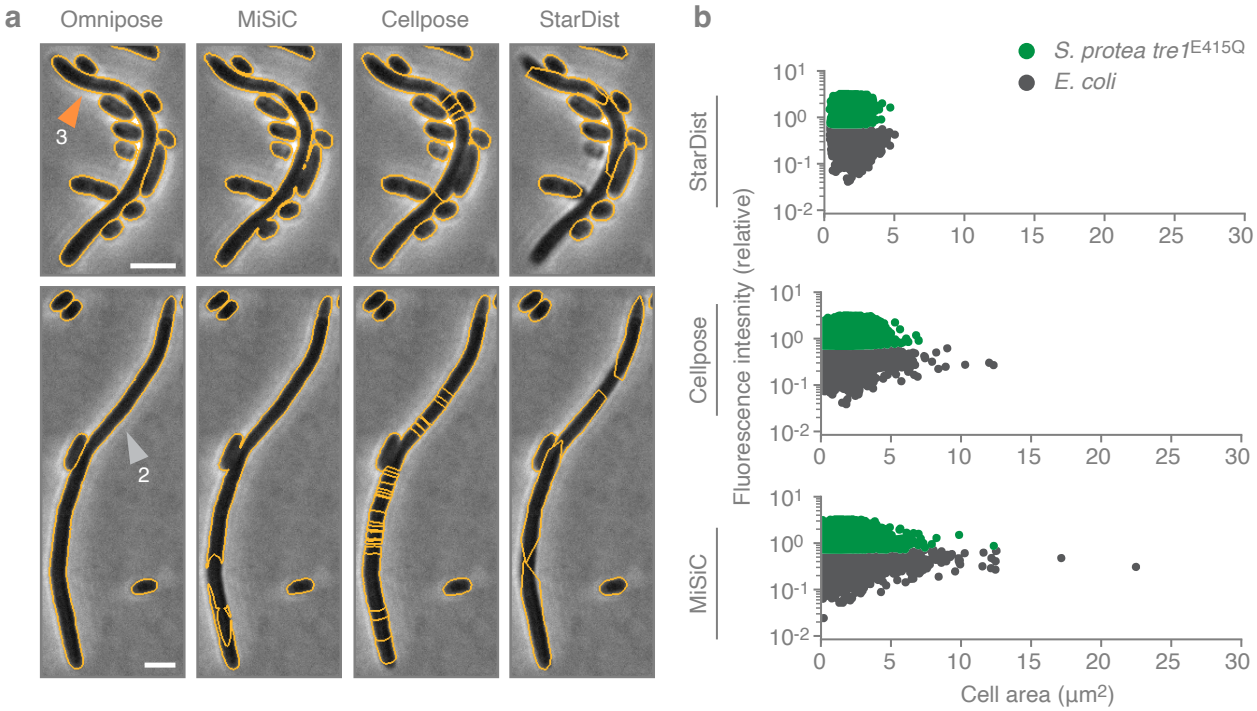

Figure 3.10: Example segmentation errors by StarDist, Cellpose, and MiSiC of *E. coli* cells undergoing intoxication by *S. proteamaculans* Tre1. (a) Examples of segmentation failures demonstrated by each method. Cells 2 and 3 indicated with orange and gray arrows are the reference cells highlighted in Fig. 3.9a,d. Scale bars are 1 µm. (b) Control populations segmented by StarDist, Cellpose, and MiSiC. Notably, Cellpose and MiSiC exhibit an enrichment of larger cells even in the control, a consequence of both under-segmented (merged) cells as well as fragments of over-segmented large cells.

Chapter 4

AFFINITY GRAPHS AND SELF-CONTACT

Self-contact is a morphological property found in many natural structures. In bacteria, this is commonly observed in cells that are extended and flexible enough to fold onto themselves (*e.g.*, Fig. 1.2 and Fig. 3.4b). In rigid cells, budding and branching growth may also allow for disjoint sections of the cell exterior to be close contact. Even more common is the formation of new cellular boundaries during division, in which two distinct layers of external cell material are necessarily grown in contact and yet remain part of the same cell until division is complete.

But what does segmentation mean in the context of self-contact? I first became deeply intrigued by this question early on in development of the `bact_phase` dataset, where I found that I had to exclude all examples of self-contact from the training set. I left this problem unresolved until I realized that self-contact is an inherent property in the segmentation of time lapses, in which cell lineages are most fundamentally represented by a single, branched, self-contacting spacetime tree (see Chapter 5). But to use Omnipose for segmenting these structures, we require a method for representing ground-truth labels from which accurate (boundary-respecting) distance and flow field may be derived. Just as importantly, the output of Omnipose would also need to be in such a form on order to do the kind of boundary-dependent analyses of single cells.

In the following chapter, Section 4.1 and Section 4.2 recapitulate Section 1.1 from the point of view of discovering affinity graphs as the answer to this important question. The remainder of this chapter describes the development of Omnipose to produce affinity segmentation.

4.1 The hierarchy of segmentation encoding

At first order, we have what is known as *semantic segmentation*, in which each pixel is assigned either the value 0 if it is background (the growth substrate) or the value 1 if it is foreground (cells). However, this binary map cannot be used to distinguish between cells in contact because the pixels of all cells are assigned the same value. To get around this, semantic segmentation algorithms are designed to avoid cell contact by leaving a boundary of 0s around each cell. This compromise results in masks that very poorly approximate the true size and shape of cells.

At second order, we have what is known as *instance segmentation*, in which background pixels are assigned the value 0 and the pixels of each object (an instance of the foreground class, *i.e.*, any cell) are assigned a unique integer label. Known as a label matrix, this integer map can be used to distinguish between two cells in contact because the pixels belonging to either cell are assigned distinct values (*e.g.*, 43 and 71). This transition between labels is the only way to define cell boundaries, and therefore instance labels cannot encode boundaries within or between disjoint parts of a given object.

At third order, we have what we will call *affinity segmentation*, in which the fundamental segmentation encoding is an *affinity graph*. Each node in the graph represents an image pixel, and edges between nodes represent connectivity between pixels in the image (here we only consider connections between pixels that are adjacent within the image, and this form of affinity graph is sometimes referred to as an *adjacency list*). Each cell in this representation is therefore a connected component of the graph, *i.e.*, a disjoint cluster of nodes linked by edges. In fact, connected components labeling is a common technique in traditional image segmentation, typically used to derive label matrices from binarized images in which foreground objects are separated by one or more background pixels. In that context, foreground pixels are said to be connected if they share an edge and/or a vertex (this is called 1 vs 2 connectivity in Python and 4 vs 8 connectivity in MATLAB). Once all connected

components are identified, the pixels belonging to each component is assigned a unique integer to form the traditional instance segmentation label matrix.

Within the foreground class, pixels may be classified as either internal or boundary. Internal pixels are those that are connected to all neighboring pixels (in 2D, either the 4 cardinal neighbors sharing an edge or all 8 neighbors sharing either an edge or vertex). Boundary pixels are those with fewer connections than internal pixels. The affinity graph is therefore constructed (sometimes implicitly) whenever boundaries are to be derived from an existing segmentation. When starting with a semantic segmentation, all foreground pixels are defined as connected to each other and disconnected from background. Thus, a binary map cannot resolve boundaries between cells in contact. Using an instance segmentation, we can define pixel connectivity by whether or not two pixels share the same label, and this can resolve boundaries between cells. However, pixels belonging to self-contact boundaries share the same label and thus are not detectable from a label matrix alone.

The affinity graph is therefore the most information-dense and general encoding of image segmentation, capable of describing boundaries regardless of whether those boundaries are part of the same cell or two different cells. Although direct construction of affinity graphs via traditional or ML approaches have been attempted [55], the most successful segmentation algorithms to date do not generate an affinity graph as an intermediate output. These include Cellpose and Omnipose [9, 56].

4.2 Boundaries are not encoded by standard instance labels

As indicated in prior work, Omnipose models accurately predict flow fields at all cell interfaces, even in instances of self-contact (Fig. 4.1a,c) [56]. This behavior is apparent in all of our models published to-date, which are not trained on datasets containing examples of self-contact. This is evidence that an Omnipose model makes its predictions based on local

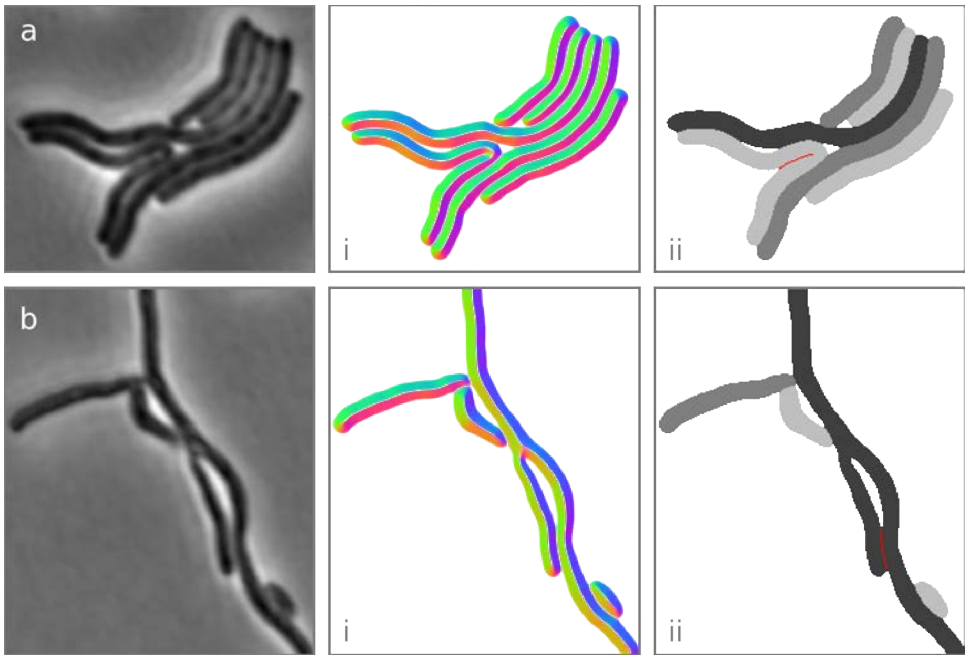

Figure 4.1: Omnipose already predicts accurate flows in instances of self-contact. (a) *S. flexneri* grown with $10\,\mu g\,mL^{-1}$ cephalexin. (b) *S. pristinaespiralis* with branching hyphae. (i) Omnipose flow predictions for images (a,b). (ii) Mask reconstructions using pixel clustering. Missing boundaries marked in red.

boundary signals within an image, a property that generalizes to any boundaries in contact whether they belong to the same cell or distinct cells.

Despite these boundary-aware predictions, the original mask reconstruction algorithm of Omnipose returns only a label matrix and therefore cannot represent self-contact boundaries (Fig. 4.1c). This is because labels are determined by spatial clustering of pixels after Euler integration using DBSCAN, and thus an affinity graph is never defined (Fig. 3.3c) [48]. Other approaches for mask reconstruction based on DNN outputs likewise do not construct an affinity graph before generating a label matrix. Approaches that directly predict pixel affinities using a DNN suffer from a lack of error correcting schemes and have therefore not risen to the same level of performance as approaches like Cellpose, and later, Omnipose [9, 55, 56].

4.3 Affinity graphs can be derived from net pixel displacement

Two key observations led us to construct an affinity graph from the existing Omnipose segmentation pipeline. First is that the flow field of neighboring pixels is roughly parallel when those pixels should be considered connected, *i.e.*, when no boundary should exist between them. This property is intrinsically linked to the definition and calculation of the flow field (Chapter 7). Second is that the integrated pixel trajectories follow this pattern while correcting for local prediction inaccuracies. Therefore, pixel connectivity can be established by thresholding the angle between displacement vectors of adjacent pixels. Pixels whose displacement angles are less than 90 degrees are considered connected, and anything greater is considered disconnected (Fig. 4.2c).

This metric for connectivity allows us to define an affinity graph for the image (Fig. 4.2d). From this affinity graph, we may then extract both the boundary pixels and the instance labels by connected components. Excitingly, this operation is faster on average than the previous segmentation approach while giving much richer information about how pixels are

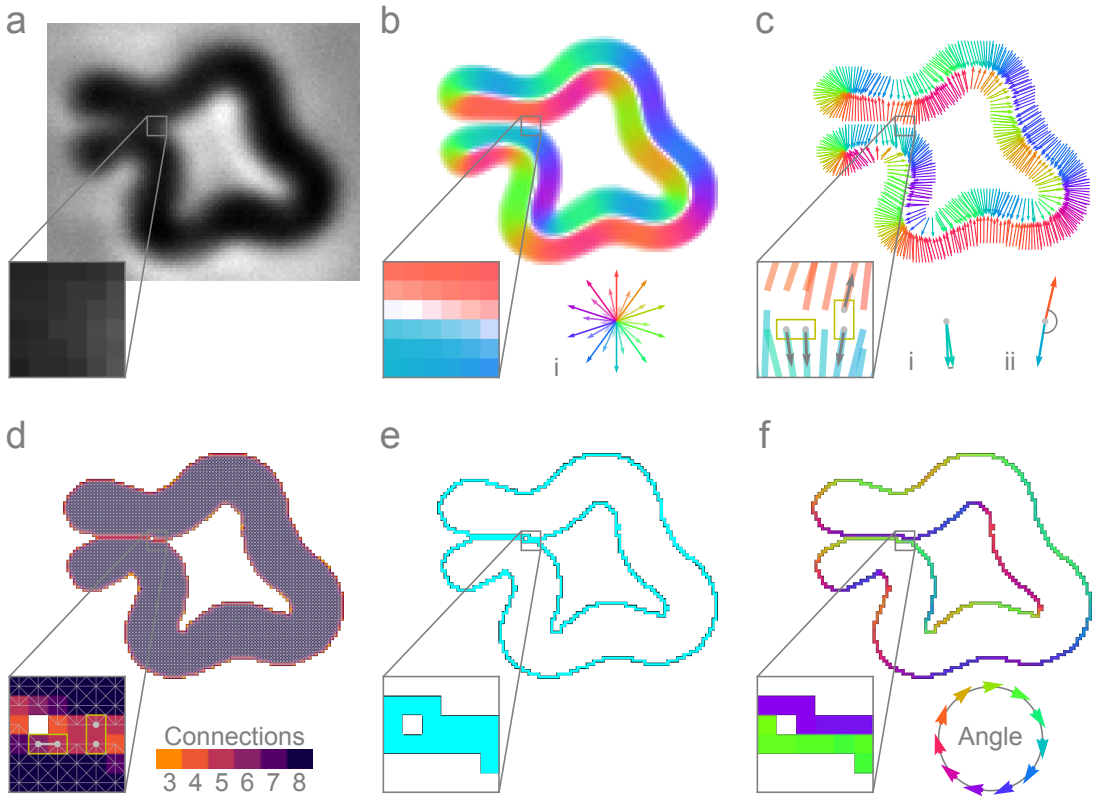

Figure 4.2: Pixel displacement defines the affinity graph, and the affinity graph defines boundaries. (a) Phase contrast image of self-contacting *H. pylori* cell treated with aztreonam. Scale bar 1 µm. (b) `bact_phase_omni` flow field predictions; phase key (i). (c) Example displacements for connected pixels (i) and disconnected pixels (ii). Connection is determined by an angle threshold. (d) Affinity graph. Lines represent connections between adjacent pixels. Heat map represents the total number of connections with adjacent pixels. (e) Boundary map. Boundary pixels are those connected to fewer than all neighbors. (f) parametrized contour; color indicates direction (i).

connected within objects. Specifically, we see that self-contact boundaries within can indeed be represented (Fig. 4.2e). The affinity graph also contains the required information to parametrize cell boundaries (Fig. 4.2f).

4.4 Euler integration error-corrects DNN flow predictions

Although we hoped that we could avoid Euler integration altogether (saving computation time) by using the angles between the raw flow field at each pixel to assign pixel connectivity, we found that this was unreliable at boundaries. This is because the raw flow field may have occasional single-pixel directional errors that cause its angle with neighbors on both sides of the boundary to be below the connectivity threshold (Fig. 4.3b,c). In this instance, thresholding relative flow angles is analogous to thresholding the cell interior and boundary fields of classic U-nets, in that small errors in local predictions can entirely invalidate the global segmentation result [9, 57]. This again leads us to conclude that use of DNN outputs without error correction of some kind inevitably leads to segmentation errors. In the case of vector field flow output, Euler integration appears to be a simple and sufficient means to error-correct the predictions.

Errors in the flow field predictions can be understood as an inevitable consequence of loss functions based on mean-squared error (MSE), in which near-zero predictions are favored compared to large, incorrect predictions. This is particularly problematic in regions where the prediction must take on one of two highly opposed values, such as diverging vectors on either side of a boundary. Because the flow field is comprised of N distinct components in N dimensions, low-value predictions tend to occur independently along each axis. As scaling one component of a vector alone will change its direction, this can result in flow field predictions that do not point toward the local skeleton of any cell. Despite this, Euler integration allows for adjacent flow predictions to correct the trajectory of an errant pixel and guide it away from the boundary with a sufficient number of iterations (Fig. 4.3).

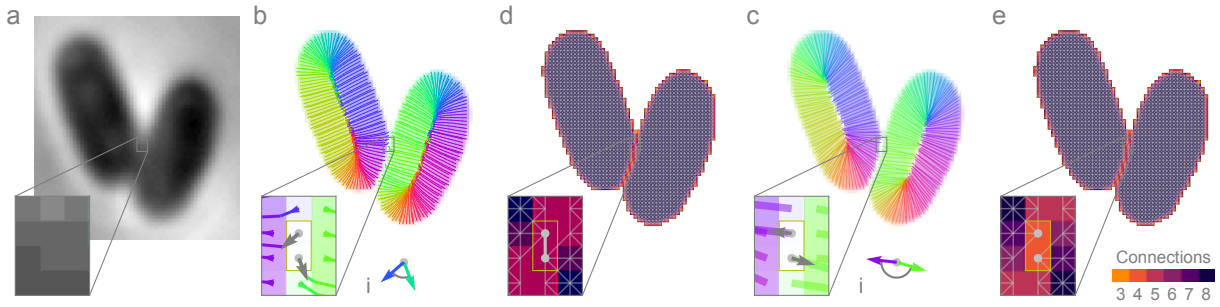

Figure 4.3: Euler integration is an error-correcting mechanism for flow-derived connectivity. (a) Phase contrast image of *E. coli* cells in contact. (b) Pixel trajectories at a boundary interface. Poor prediction at this interface allows pixels from adjacent cells to be below connectivity angle cutoff (i). (c) Affinity graph defined by pixel displacement from 1 integration step). This one errors corresponds to under-segmentation (cells remain connected). (d) Net displacement after 3 integration steps corrects local errors in flow and pixel displacement. Angle between boundary pixels is now above the cutoff (i). (e) Affinity graph defined by pixel displacement after 3 integration steps.

4.5 Sine squared loss alleviates interpolation at cell boundaries

Despite this error correction scheme, we nonetheless sought to improve flow prediction quality at cell boundaries. Prediction quality on a given dataset is largely determined by the loss function, *i.e.*, by the energy landscape in which the network parameters are being optimized. To this end, we devised a novel loss function, sine squared loss, defined as

$$\mathrm{SSL}(p, \hat{p}) = \frac{1}{n}\sum_{i=1}^{n} \sin^2 \theta_i = \frac{1}{n}\sum_{i=1}^{n}(1 - \cos^2 \theta_i) = \frac{1}{n}\sum_{i=1}^{n}(1 - \frac{\mathbf{p}_i \cdot \hat{\mathbf{p}}_i}{\|\mathbf{p}_i\|\|\hat{\mathbf{p}}_i\|})$$

where n is the number of samples (pixels), p_i are predictions, $\hat{p}_i$ are ground truth, and θ_i are angles between predicted and ground-truth vectors. This function captures the intuition that predicted vectors should always be parallel or antiparallel to the ground truth, but never orthogonal. Because we have heavy warping and subsequent nearest-neighbor interpolation of labels during training, there are often label artifacts at cell boundaries at the scale of

single pixels. Thus, we are more interested in demanding that the flow field points into *any* cell at an interface rather than into a *particular* cell, as MSE would demand.

Whereas MSE drives the network to predict smooth changes in the flow field across an interface (necessarily reaching zero magnitude between cells), SSL drives the network to predict nonzero *opposing* vectors at an interface. We found that SSL in combination with the norm loss and reduced contributions from MSE allows the network to predict discontinuous, constant-magnitude, opposing flow field components across cell boundaries (Fig. 4.4).*†

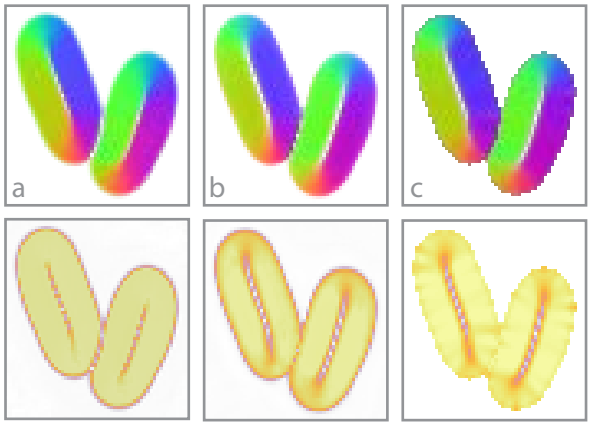

Figure 4.4: Sine squared loss remediates errors in magnitude and direction at cell boundaries. Top row: flow field phase plot. Bottom row: flow field magnitude. (a) `bact_phase_omni`, trained without SSL, predicts flows that go to zero at the cell interface. Low-magnitude predictions at the interface also tend to have a component parallel to the boundary (see Fig. 4.3). (b) `bact_phase_omni_2`, trained with SSL, predictions flows that are close to ground-truth magnitude. Low-magnitude predictions at the interface now point orthogonal to the cell interface. (c) Ground truth flow field. Gradient computation is now designed for the magnitude to fall off at the cell skeleton but not at cell boundaries.

*MSE on flow components is still useful for initial convergence.

†Note on Fig. 4.4: `bact_phase_omni` was trained on normalized flows. However, unlike the Cellpose heat-derived field that required normalization, we realized that the distance field being defined by the Eikonal equation means that its gradient should already have unit magnitude everywhere. This is not precisely the case due to way we iteratively compute the distance field, but by constructing a custom gradient operator, we were able to compute an accurate gradient without post-hoc normalization. The zero magnitude at the skeleton is a bonus, as it improves both training and mask reconstruction.

4.6 Self-contact requires a modified annotation technique

We have so far shown that existing models trained on examples without self-contact boundaries can generalize to cases of self-contact, and we have further demonstrated that Omnipose can now detect such boundaries by constructing an affinity graph based on integrated pixel trajectories. However, all models will eventually fail when applied to data sufficiently unlike the training set. For example, *E. coli* grown on cephalexin produces extremely elongated cells with thin and phase-light regions (Fig. 4.5a). We hypothesize that `bact_phase_omni` model interprets the morphology of tight turns and the sharp optical transition between dark and light (where the chromosome is excluded) as a boundary between two cells, leading to over-segmentation (Fig. 4.5b). To accurately segment images like these, we must retrain the model on a dataset that includes images with these features. Because these features only exist in such systems with extreme filamentation and self-contact (tight turns necessarily imply self-contact), we next sought a method to annotate such cells and compute accurate ground-truth distance and flow fields from those annotations.

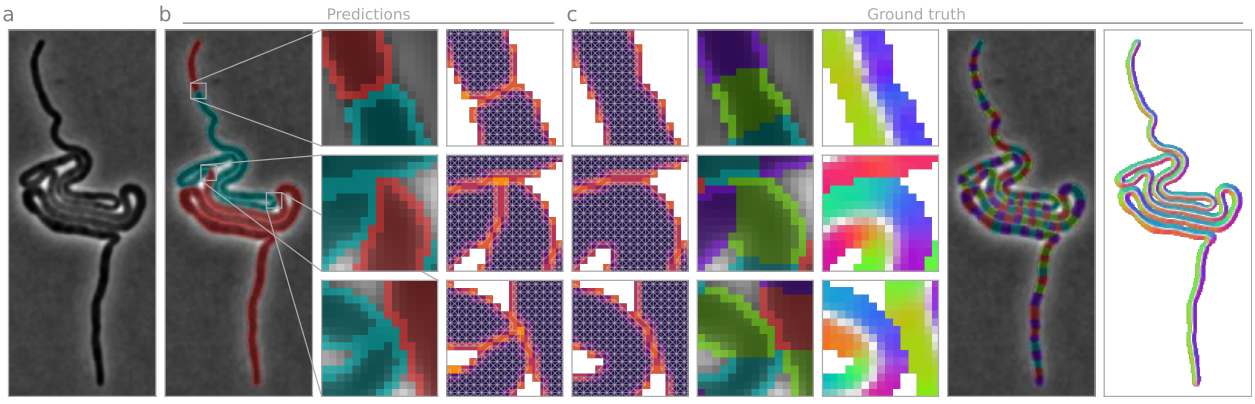

Figure 4.5: Multi-label annotations on extended, self-contacting cells. (a) Representative *E. coli* BW27783 cell grown 10 µg mL^{-1} cephalexin for 9.5 hours on an agar pad. (b) Predicted flow and segmentation using `bact_phase_omni` model. (c) Multi-label representation of the cell. (i) Many unique labels make up one cell. (ii) Connectivity across labels is defined in a link matrix. (iii) Affinity graph computed from labels and links. (iv) Distance and flow computed from affinity graph.

The need to train models on systems with self-contact boundaries poses yet another distinct challenge. Ordinary image annotation using instance labels is already a difficult and tedious endeavor [58]. At its most efficient, this process involves manual correction of masks made by a somewhat accurate segmentation algorithm by splitting or merging instance labels. However, as aforementioned, we cannot use instance labels to represent self-contact boundaries (Fig. 1.2). Our initial solution was to use an additional annotation channel (or layer) for binary boundaries. One may incorporate such a layer in existing GUIs such as Napari and Photoshop, but we ultimately abandoned this methodology for three reasons. First is that the instance mask and boundary layers are decoupled, requiring the human annotator to constantly toggle between layers to ensure that the boundary map corresponds exactly to the instance map pixel-for-pixel. Second is that the boundary map is ultimately used to define an affinity graph to compute distance and flow fields, requiring it to be formatted as precisely 2px-thick at all interfaces. This is not only difficult to do manually, but it also cannot be used to define an affinity graph for thin cells. Third is that image augmentations during training require labels to be well-defined after transformations using nearest-neighbor interpolation, but 1–2 px wide features are irreconcilably corrupted during the required affine transformations (Chapter 7).

Our solution was to define multi-label instance maps, in which a single object is split into multiple large ($\geq$9 px^2) integer labels and linked using a separate dictionary saved as a plain text file. Pixels with dissimilar labels are now connected if these labels are listed as a pair in the link file, thereby allowing for the generation of ground-truth distance and flow fields for training a new model (Fig. 4.5c). In the context of self-contact boundaries, at least three labels are required: two unlinked labels on either side of the boundary, and one at the end of the boundary that is linked to both labels on either side. Multi-label instance maps can be rapidly created by human annotators for any cell type. As described in prior work, annotation can be dramatically accelerated by correcting flawed DNN segmentation [56]. This is also true in the context of self-contact boundaries, but generating multi-label instance maps (even from

affinity graphs) is highly nontrivial. We developed custom pipelines for generating multi-label annotations for human validation for the two distinct cels morphologies presented in this study. In both cases, the lack of thin features (<3 px wide) guarantees the existence of interior cell pixels throughout the cell body, thereby allowing us to use multi-label boundaries linked by interior pixels. This simplification allows for automatic post-processing of single-layer annotations into true multi-label instance maps and link files without further human input (Chapter 7).

4.7 Affinity segmentation enables the analysis of elongated, self-contacting cell morphologies

Having built the infrastructure to train on self-contact data, we next annotated a full time lapse a single *E. coli* cell growing to 38× its original size in 10 µg mL^{-1} cephalexin (Fig. 4.5). We then added one third of this data in the form of composite frames to the BPCIS dataset on which the `bact_phase_omni` model was previously trained. We found that the resulting model `bact_phase_omni_2` performs as well as `bact_phase_omni` while also accurately detecting boundaries across a range of elongated, self-contacting cell morphologies (Fig. 4.6).

Ordinary rod-shaped cells can be automatically registered to a standard rod shape, effectively mapping the captured data to a defined local coordinate system. This normalized data can then be averaged across a population of cells (known as a consensus image). Registration to a known orientation also trivially allows for the projection of the cell data onto its long axis (now taken to be horizontal), reducing the spatial dimension and allowing many such projections to be stacked over time to form a composite image (known as a kymograph). Consensus images and kymographs are invaluable tools for understanding distributions of biomolecules and cellular structures in space and time, but all existing tools for creating them break down for long, curved, or self-contacting cells.

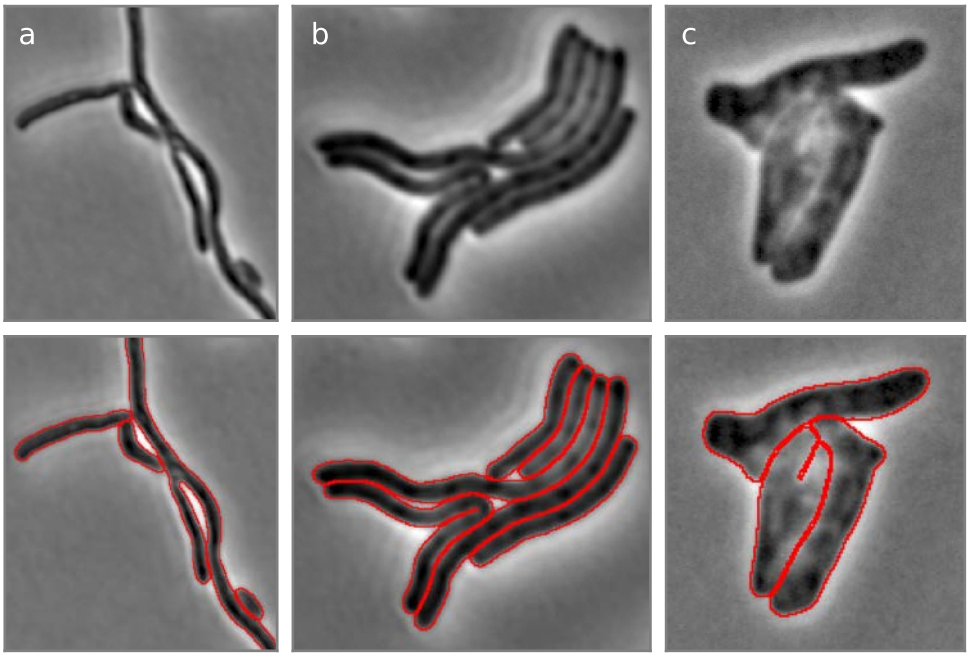

Figure 4.6: Affinity segmentation on elongated, self-contacting cells. In addition to ordinary cell morphologies, Omnipose now accurately segments (a) branching hyphae in *S. pristinaespiralis*, (b) invaginated *S. flexneri* ($10\,\mu g\,mL^{-1}$ cephalexin), (c) legs in *E. coli* CS703-1 ($1\,\mu g\,mL^{-1}$ aztreonam).

The affinity graph allows us to generalize the standard approaches measuring and aggregating single-cell data in elongated, curved, and self-contacting cells. To illustrate this, we utilized a HupB:YFP fusion to visualize the chromosome within replicating (but not dividing) *E. coli* DY330 growing on a $1\,\mu g\,mL^{-1}$ aztreonam, 3% agarose LB pad (Fig. 4.7a). The affinity graph generated by Omnipose allows us to define an exact cell skeleton, distance field, and flow field, which we then use to project the image data onto the cell skeleton for each time point, thereby generating a kymograph (Fig. 4.7b). This approach can be used for any cell shape with a non-branching skeleton.*

*2D kymographs are not applicable to cell shapes that cannot be reduced to 1D, as the second dimension is used for time. However, the affinity graph contains the require information to construct a local coordinate system and register to a standard shape, thereby making consensus images possible even when self-contact is involved.

In this instance, we observed six distinct phases in cell growth as measured by the relative area over time (Fig. 4.7c,d). Phases 3 and 4 follow a sharp contraction of the cell. Although such a measurement could have been made by area alone, the affinity graph allowed us to further decompose this measurement into the average width and length (Fig. 4.7e,f). We found that the length of the cell grows exponentially without interruption, and it is instead the width of the cell that accounts for the reduction in cell area.

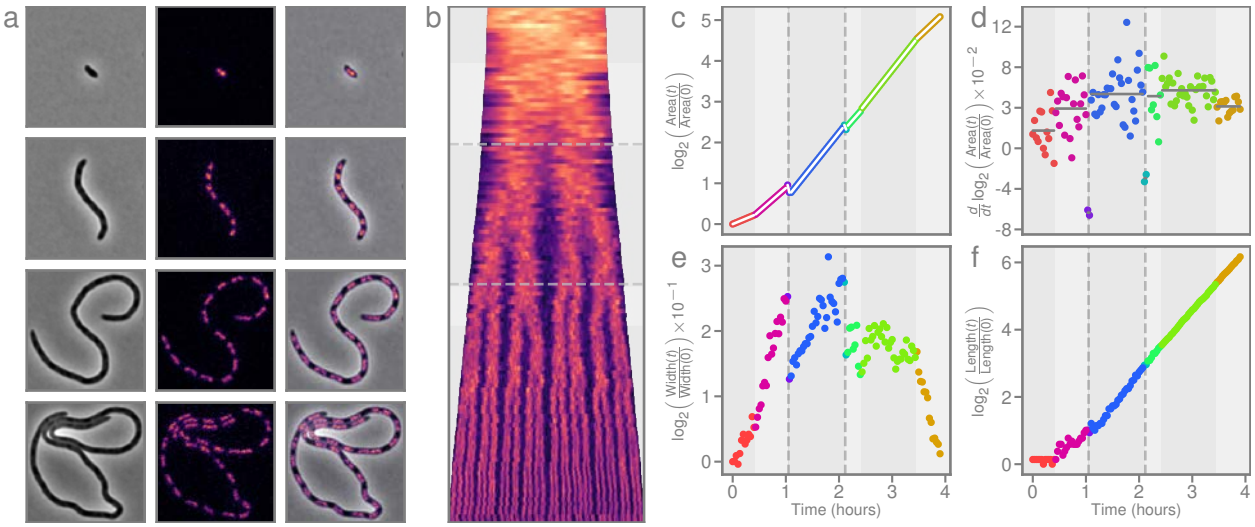

Figure 4.7: HupB localization and sensitive morphology characterization in filamentous, coiled *E. coli*. (a) Cell treated with $1\,\mu g\,mL^{-1}$ aztreonam at $t = 11, 71, 102, 118$ minutes. (b) Kymograph (distance-weighted projection onto skeleton) of HupB signal. (c) Growth measured by the relative change in area. Six distinct growth phases are present (gray regions, dashed lines indicate sudden shrinkage events). (d) Growth phases are determined by using a Decision Tree Regressor on the measured growth rate.

4.8 Affinity segmentation enables septum tracking in dividing cells

During division, two new cell boundaries are synthesized in contact (one for each daughter cell). The developing septum within the mother cell represents a nearly universal instance of

self-contact boundaries. However, this structure is not directly observable in most bacteria by phase contrast microscopy, where the signature of division is only detectable by a developing cusp in the cell profile. Fluorescence imaging is therefore the modality of choice for studying the cell membrane.

The bacterium *Deinococcus radiodurans* is particularly amenable to fluorescent membrane imaging due to its size (overcoming diffraction limits) and its relative tolerance to the many excitation exposures required for time lapses [59]. Previous studies have demonstrated that the cell cycle can be tracked by the unique morphology displayed by the developing cell septum stained with Nile Red [59]. However, prior quantitative analyses were highly limited in throughput due to the lack of a trained DNN segmentation model for this kind of data. Using our new multi-label annotation technique, we were able to train a new omnipose model (`drad_fluor_omni2`) that successfully recognizes all cell boundaries, including newly formed boundaries at the septum. The boundary detection is extremely precise, allowing up to single-pixel openings in the septum just before division is complete (Fig. 4.8).

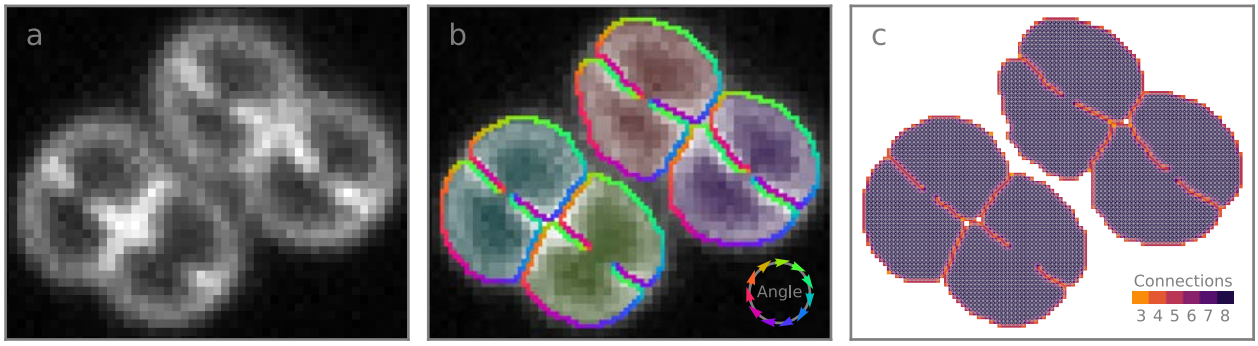

Figure 4.8: Sensitive detection of cell septum during division. (a) Representative *D. radiodurans* microcolony visualized with Nile Red. (b) Omnipose segmentation with overlaid parametrized boundaries. (c) The underlying affinity graph for this segmentation.

Chapter 5

SPACETIME SEGMENTATION

We have thus far shown that Omnipose can be used to segment extended objects in 3D. We have also shown that Omnipose can segment objects exhibiting self-contact, meaning that boundaries are represented within single objects. These capabilities are exactly what is needed for segmenting dividing cells in 3D (2D+T) spacetime, in which cell lineages form extended, branched, self-contacting volumes. This method of segmentation will solve multiple problems in the traditional time lapse processing pipeline.

5.1 The problem of cell tracking

The goal for time lapse segmentation is inherently three-dimensional, as we require each cell to ultimately receive a unique identifier for every pixel and time point at which it has been imaged. This is commonly referred to as a *Cell ID*, but I find the term *spacetime label* to be a more natural and evocative of the underlying geometry.

The task of generating spacetime labels has always been broken down into two stages. First, a time lapse[†] is segmented frame-by-frame, where the segmentation at t does not influence the segmentation at $t+1$. These per-frame labels are uncorrelated over time and are effectively random numerical values. The second step is to link the uncorrelated labels for each single cell over time, called *cell tracking*.

Cell tracking algorithms construct a loss function from properties like frame-to-frame mask overlap and momentum. The minimization to this loss function determines the most likely

[†]with in-plane movement removed by frame-to-frame image registration, also called "image alignment" or "stabilization"

tracking result, *i.e.*, the most probable set of all frame-to-frame label links. However, errors in segmentation inevitably corrupt all tracking algorithms.*

5.2 The problem of cell division

In terms of frame links, binary cell fission is detected when a given frame label at time t is linked to two frame labels at time $t+1$.† The cell at t is known as the *mother*, and the two cells at time $t+1$ are known as the *daughters*. The set of all mothers and daughters forms the lineage tree. The correct determination of cell lineages is a crucial component of analyzing and interpreting results in live bacterial microscopy.

Cellular division is not instantaneous, however, and so a continuum of intermediate states can be captured under the microscope (Fig. 5.1a,b). These intermediate, ambiguous division states result in temporally incoherent decisions around cell division, wherein a mother cell may be whole at time t_1, divided into two daughters at time t_1+1, and then rejoined into the mother cell at time $t_2 > t_1+1$ (Fig. 5.1d, Fig. 5.2c). At high frame rates, a model may "change its mind" several times during a division event (splitting and joining a cell many times) until finally reaching a stable prediction about cell division. These temporally incoherent predictions pose a significant challenge to existing tracking algorithms and bacterial analysis pipelines.

No current 2D image segmentation model is built to utilize any temporal information in its predictions.‡ Human beings can and do make similar errors when given only a single frame of a time lapse and asked to decide whether or not a dividing cell has completed division,

---

*It is possible to design ad-hoc approaches that fix specific kinds of segmentation errors that are detectable in the tracking pipeline, but these constitute a small subset of common issues that arise in time-lapse microscopy.

†Other forms of division, occasionally exhibited by bacteria, can result in more than two daughter cells.

‡At least, I am not aware of any implementation that leverages temporal information for dense object segmentation. I could see region-based approaches doing this, but as we saw with Mask R-CNN, those are not applicable to microscopy images.

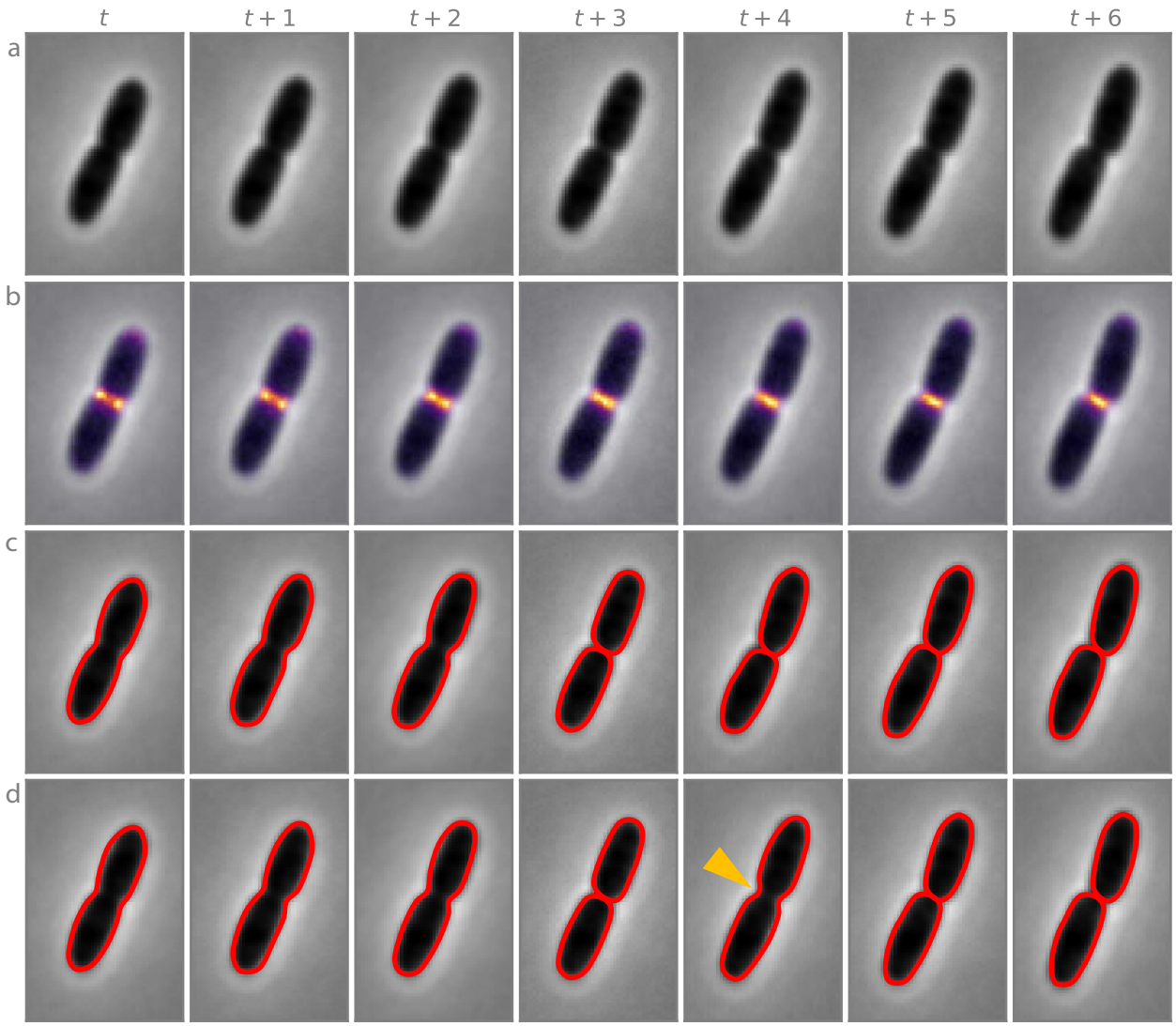

Figure 5.1: Intermediate division states and segmentation errors. (a) Representative *E. coli* division over 2 minute intervals. Division state is difficult to determine from phase contrast alone (b) Pal-mCherry signal overlay may indicate true division as early as $t + 3$ and as late as $t + 5$. (c) Possible correct segmentation. (d) Example of mother-daughter cell merging. Daughter cells at $t + 3$, merged at $t + 4$, and divided again at $t + 5$.

and so it is not unexpected that neural networks would make the same mistakes given the same limited information.

5.3 The solution in the high-frame-rate regime

The solution to these issues, as I see it, is to sidestep the problem of cell tracking/linking and generate spacetime labels directly. From this perspective, entire cell lineages are the fundamental 3D objects that are to be segmented from 2D+T time-lapse volumes. The affinity graph representing this segmentation groups all cells in a lineage together, thereby removing the need for a separate step of cell linking. This graph can be sliced at each time point to produce per-frame segmentations for making measurements on individual cells.

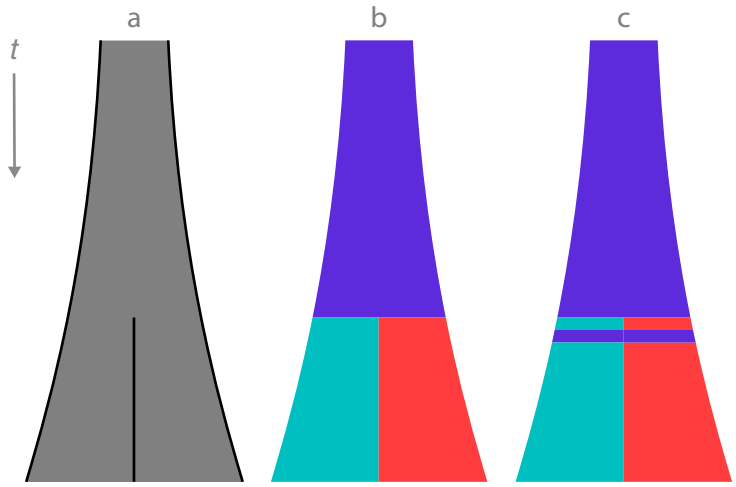

Figure 5.2: Cell segmentation in 3D spacetime. (a) Kymograph schematic of cell growth and division. A mother cell grows until the new cell wall develops and daughter cells are born. Cell boundary indicated in black. (b) Spacetime labels for the mother and two daughter cells. (c) Schematic example of mother-daughter merging corresponding to Fig. 5.1d.

For sufficiently high frame rates (such that the sampling frequency in time is roughly equivalent to the sampling frequency in space), cellular spacetime volumes are contiguous

and pose a problem completely analogous to spatial 3D segmentation. In this regime, we can therefore apply the standard approach of Omnipose for training and segmentation. For much lower temporal sampling, where cells may be moving too fast as to have any overlap with themselves in the previous frame, this approach of spacetime segmentation breaks down. However, most time lapses taken for the purposes of exploring sub-cellular processes (which occur on time scales much smaller than that of cell division and movement on agarose pads) do have a high enough frame rate to make this approach feasible.*

5.4 Spacetime training data

Dividing lineages in 3D spacetime pose the same self-contact problem as folding cells in 2D space. To properly generate ground-truth predictions like distance and flow, we must split the lineage label into parts. Fortunately, the natural solution to this is just a spacetime labeling of all individual cells (mothers and daughters) (Fig. 5.2). The links in this context are just mother-daughter links; *e.g.*, cell 1 is the mother of 2 and 3, 2 is the mother of 4 and 5, and so on.†

5.5 Extracting tracked cells from lineage affinity graphs

Typical analysis pipelines are built to expect frame-by-frame cell masks and frame-to-frame mask links. Thus, we must be able to generate these outputs from the more fundamental affinity graph in 3D spacetime. This is achieved by simply deleting all edges in the affinity graph that connect voxels at time t to $t \pm 1$. The remaining graph now consists of disjoint

*Cells that are more sensitive to frame rate due to phototoxicity or photobleaching may require fluorescence to be less frequently acquired than phase contrast.

†This simplification only holds for dividing cells that have no spatial self-contact, of course. It is possible to train a spacetime model on time lapses of such cells, but the training data would be much more difficult to make because there would be many more sub-labels and the links would be in all 3 dimensions instead of just time.

2D slices of the affinity graph for each time point. The subgraphs corresponding to each cell then receive a unique identifier using connected components labeling.

With these frame-by-frame labels, we then use the temporal edges of the affinity graph to generate frame-to-frame links. By brute force, we can use our affinity graph to query each note and generate all pairs of labels for each voxel and each of its possible 26 neighbors in its $3 \times 3 \times 3$ neighborhood. This operation can be optimized somewhat by only considering connections from t to $t + 1$. After this list of frame-to-frame links is generated, we then remove the many duplicates in this list arising from frame-to-frame mask overlap.

5.6 Preliminary results

As an initial proof of concept, I curated a collection of 15 spacetime volumes for semi-manual linked-label annotation. To capture morphological diversity, I included 4 micro-colonies of *A. baylyi* essential gene mutations (DnaN, DnaA, MurA, FtsN) [60]. To capture intermediate divisions states, I included 11 microcolonies of *E. coli* captured at high frame rates.* Each was segmented with Omnipose, tracked with Trackmate or SuperSegger, remapped into spacetime labels, and then loaded into Napari to correct any errors and manually generate mother-daughter links [5, 10, 61].† These labeled and linked volumes were then used to train a new omnipose model, `bact_phase_spacetime`.

To assess performance qualitatively, we can look at the DNN output and mask reconstruction on representative *E. coli* microcolonies. We find that `bact_phase_spacetime` has temporal coherence throughout division events, solving the problem of incoherent mother-daughter merging exhibited by `bact_phase_omni` (Fig. 5.3).

---

*With a doubling time of about 30 min, I used frame rates from 30s to 2 min.

†Errors at this stage may arise both from segmentation errors as well as tracking errors. Mother-daughter links coming from a tracking algorithm cannot be trusted and must be either checked or generated manually; I found it faster to do the latter.

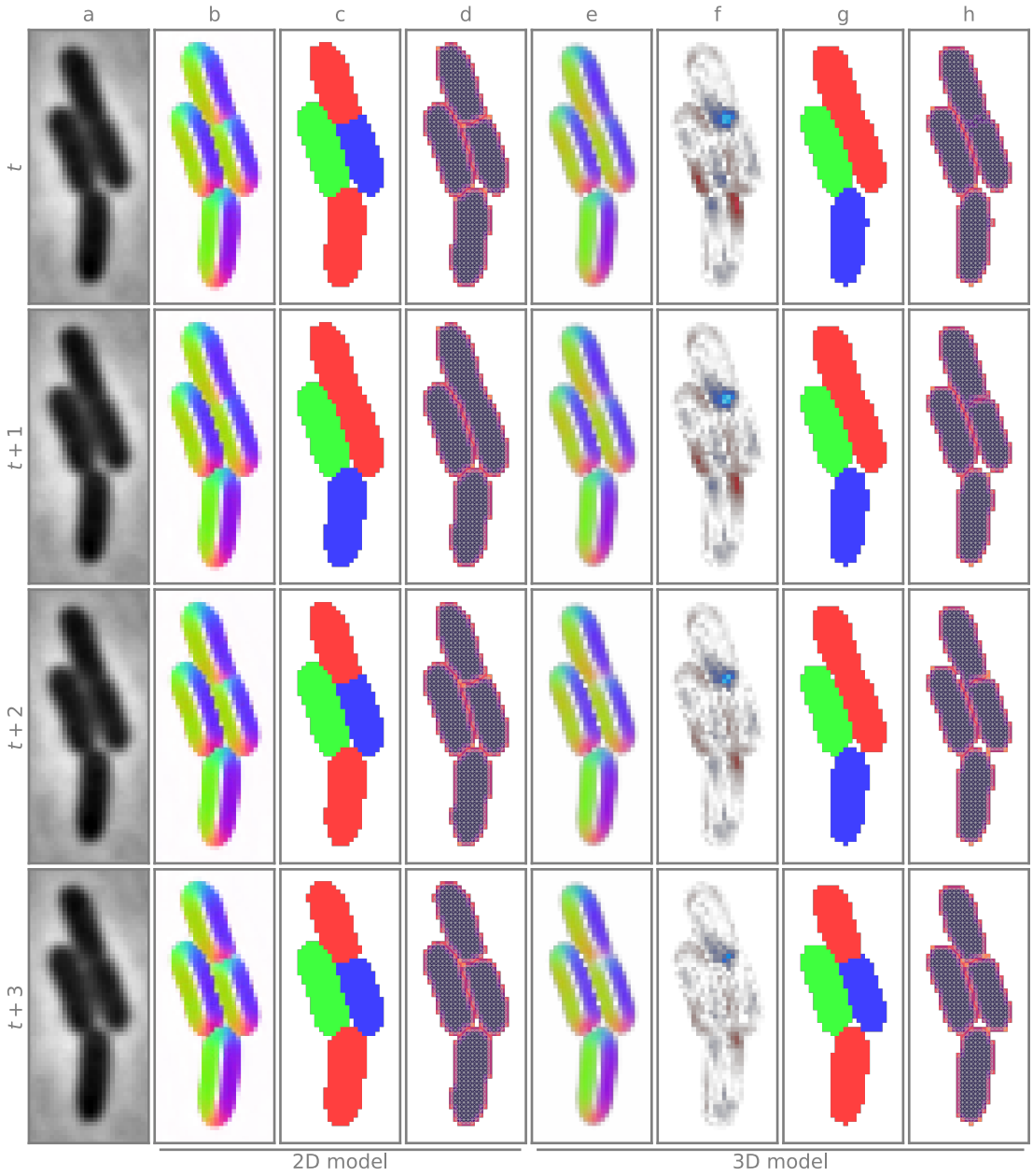

Figure 5.3: Spacetime model provides coherent segmentation through division events. (a) *E. coli* division acquired with phase contrast at 2 minute intervals. (b) `bact_phase_omni` flow predictions. (c,d) Mask and affinity predictions based on (a). Daughters at time t are merged at time t+1. (e,f) Spatial and temporal flow components from `bact_phase_spacetime` model. Positive temporal flow is red, negative is blue. (g,h) Mask and affinity predictions based on (e,f). The boundary between daughter cells forms gradually.

5.7 Future work

To make a spacetime model that is as general and applicable as our models in 2D, we must train on a much larger dataset. The poses two main challenges.

5.7.1 Physiological accuracy of training data

A spacetime model has two goals: (1) temporally coherent determination of cell boundaries, and (2) physiologically accurate determination of cell boundaries. Temporal coherence allows for sensible cell division and tracking, but accurate cell lifetime and protein localization within the cell depends on physiologically accurate boundaries both in space and in time. A model trained on temporally biased data will learn to replicate that bias (early or late divisions) whereas noise in the ground-truth data (a mix of early and late divisions) will corrupt the training process and converge to parameters that give locally averaged/blurred predictions.

The solution to this problem is to annotate cells using a clear marker for cell division. As we need to train models on high-frame-rate data, we require our marker to be highly resistant to photobleaching. To this end, I grew and imaged dozens of *E. coli* strains in the ASKA2 database for cytosol and membrane-localized protein-GFP fusions as well as strains in the Wiggins and Mougous libraries with constitutive or induced flurophore expression. Although I found two candidates for strong cytosol expression, all membrane-bound proteins in the ASKA database were not expressed at a high enough level to persist over long time lapses at high frame rates. We ultimately obtained a Pal-mCherry *E. coli* strain that was suitable [62]. Although mCherry is less ideal than GFP due to the wider diffraction pattern and therefore lower spatial resolution, the outer membrane protein Pal is one of the most highly expressed proteins in *E. coli* and so provided a sufficiently clear signal to serve as a ground-truth marker for cell division.

At this time, I have collected high-frame-rate time lapses of several hundred wildtype *E. coli* microcolonies with either bright GFP cytosol or outer-membrane mCherry fluorescence, several of which were used to train `bact_phase_spacetime`. I have also collected dozens of similar movies of antibiotic-induced *E. coli* filamentation (aztreonam, cephalexin) and rounding (A22) with GFP cytosol or FtsZ-GFP.* However, the diversity in cell morphology and internal structure remains limited in this data, as only a few strains of *E. coli* were used. A full dataset will, at minimum, require genetic manipulation of multiple species to express a bright and persistent cytosol or outer-membrane marker.

5.7.2 Human annotation bottlenecks

Although the 2D `bact_phase_omni` model can be used to ease the burden of manually segmenting each frame of a time lapse, and tracking tools like Trackmate can help link frame-by-frame labels into spacetime labels, the aforementioned mother-daughter merging problem can only be resolved by manual editing. Even in the case of perfect segmentation and self-linking into spacetime cell labels, these results must be checked manually. The number of mother-daughter links grows exponentially in time lapses covering multiple cell cycles, requiring hundreds of links to check manually in typical microcolonies (no such data is currently in my training set for this reason). Moreover, these deep microcolonies can no longer be linked by viewing them in 3D, as cells inside a microcolony are occluded by the cells at the edge.

To alleviate this bottleneck, a GUI should be developed to enable interactive editing of mother-daughter cell links.† This GUI should allow the visualization of cell lineages in an abstract lineage tree schematic as well as in the 3D spacetime label volume. The cell

*Z ring probes, such as FtsZ-GFP or ZipA-GFP, are not bright enough to be used by human annotators in these extended, high-frame-rate movies. I have not excluded the possibility that a more highly expressed target on the cell septum may exist. Future attempts to design fluorescent fusions should begin with this question.

†The best way of doing this would probably be as a Napari plugin.

occlusion problem should be handled with translucency; *i.e.*, only the spacetime cell labels of a highlighted lineage in the tree should be opaque in the volume view.

Chapter 6

IMAGE ANNOTATION

The following sections compile my strategies, rationale, and techniques for curating ground-truth datasets for image annotation.

6.1 Garbage in, garbage out

Image segmentation tools are only as good as their training data. Although large semantic and even instance segmentation datasets do exist across a variety of fields, the majority of these are of extremely low quality. This is because many popular applications like face detection do not require pixel-level accuracy in the final output. The training data, therefore, can be generated by non-experts en masse via crowdsourcing marketplaces. Errors in individual human annotations wash out in the aggregate, and any interpolation effects from the disagreements at object boundaries are considered acceptable.

For scientific applications concerned with sub-pixel localization of structures within objects tens of pixels wide, errors in ground-truth annotation cannot be tolerated. Fortunately, my work suggests that smaller, carefully curated human annotations are more effective at training accurate models than larger, hastily curated datasets. Less is more, especially if the goal is to fine-tune a model to work on a specific image set.

6.2 Dataset and image sizes

This principle applies to image size as well. Large images should be split into several small images for two important reasons. First is that only a fixed subset of the image is sampled per training epoch. With the default patch size of `(224,224)`, a full-resolution image from a microscope on the order `(2048,2048)` would take at least 84 epochs for the network to "see"

the entire image. Cropping the image instead to 16 `(512,512)` images allows the network to effectively train on the entire image in parallel.

The second benefit to splitting images for annotation is that they are less prone to error. Large images contain more cells and are fatiguing to fully and carefully annotate. Large images also natively display their full extent, where cells and possible mistakes in their annotations remain small and difficult to discern. Small images encourage human annotators to keep all cells in view at a magnification that reveals the errors to be corrected, which can then be swiftly made and saved as manageable subsets of the full image.

Ground-truth images should not only be cropped for size, but also for content. For example, all examples of self-contact must be excluded unless linked multi-labels are provided. Cell overlap must likewise be excluded, as sensible ground truth cannot be established. All cells must receive a label, but it is possible to include images with unlabeled cell debris alongside annotated cells to train a model that ignores cell debris. This strategy may not apply to excluding objects that are too cell-like (such as microfluidic chamber pillars), but this is not been tested.

Lastly, I will note that the training set should be balanced in its diversity of example images. The order of images in the training directly is randomly shuffled, but the number of times a cell type is shown to a network is directly proportional to the number of representative images that exist for it in the training set. If the goal is to train a model that performs equally well across a range of cell/image types, then roughly the same number of images per type should be in the training set.

6.3 Instance labels require fluorescent labels

As mentioned, a neural network will learn better if the examples it is shown are annotated consistently. However, even a consistent set of ground-truth labels may be physiologically inaccurate. This typically manifests as a bias in cell dilation or contraction. Again, while

most image segmentation tasks can ignore a bias of a pixel or two, we would like to avoid this when generating masks that fundamentally determine the meaning of the results we measure via image cytometry.

The solution here is to use fluorescent cell membrane labels. The Pal protein in *E. coli* is excellent, but a high-copy inner-membrane protein would be even better. At worst, an outer-membrane dye would be better than nothing.

6.4 The four-color theorem

All existing software for image annotations uses the same deeply flawed approach for visualizing instance labels: one unique color per cell mask overlaid onto the source (typically grayscale) image. If there are many cells in the image, however, we quickly run out of unique colors to distinguish adjacent cells (Fig. 6.1a). To solve this adjacency issue, colors are randomly shuffled. This somewhat mitigates the problem, but some adjacent cells will inevitably receive a similar color and become very difficult to distinguish (Fig. 6.1b).

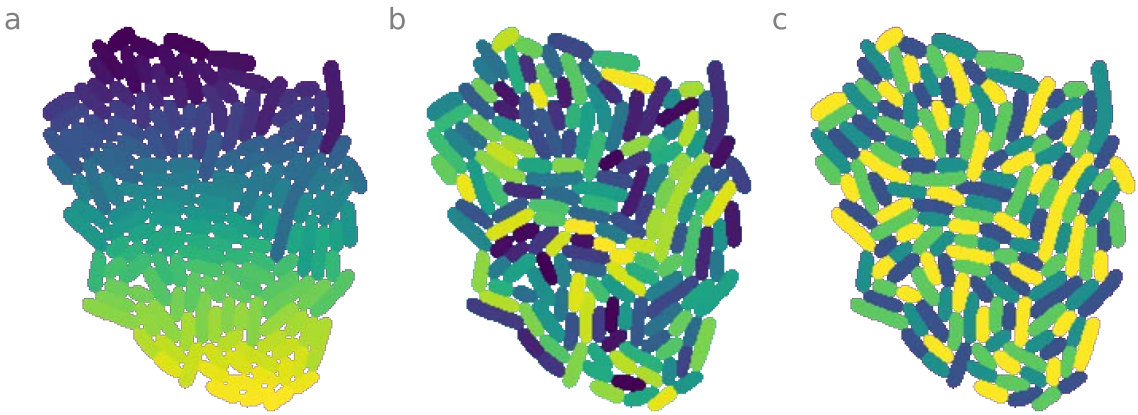

Figure 6.1: The 4-color theorem in action. (a) Perceptually uniform color map (`viridis`) applied to segmentation of an *E. coli* microcolony. There are 161 cells and therefore 161 unique colors. Instance labels typically increment from the top left corner, and so spatially adjacent labels tend to be numerically adjacent. (b) Randomly shuffled colors show distinction between some cells, but not others. (c) 4-color mapping reveals all cells with no ambiguity.

This similarity between colors makes it very easy for fatigued human annotators to select an incorrect (yet visually indistinguishable) color for editing a particular cell. Many publicly available cell segmentation datasets, particularly those with large numbers of cells per FoV, contain misshapen and punctured cell masks that I suspect could only have gone unnoticed due to this issue with label coloring.

The solution to this problem goes back to 1852, when Francis Gurthrie conjectured that only four colors are needed to fill all regions of a map such that no two regions sharing a common boundary share the same color. This was later proven in 1976 by Kenneth Appel and Wolfgang Haken followed by additional proofs in 1997 and 2005 [63, 64]. However, it appears that no software packages (at least in Python and MATLAB) have been developed to remap instance labels to only 4 values.

Soon after beginning work on generating a ground-truth dataset, I realized the need for such a tool and developed my own Python package to "4-color" instance labels. My method is stochastic* and opts to find a fast solution using 5 (or occasionally more) colors instead of attempting more permutations to achieve a coloring with 4. The method works in 3D as well, where no analogous theorem exists, and it achieves 6-7 colors for the bounded and typically convex volumes of cells in 3D and 2D+T.

This reduction from N cells to $n \ll N$ colors allows for a very limited and therefore distinguishable set colors to be used when viewing and manually annotating cells (Fig. 6.1c). It is also very handy for visualizing segmentation results and instance labels in general.

6.5 Human in the loop

Correcting annotations is often much faster than generating them from scratch. I therefore begin annotations either by (a) choosing an existing Omnipose model trained on similar data, or (b) annotating a single, small crop of the data to train a working model. Because poor

*Random seed setting seeds reproducibility, however.

flow predictions at boundaries tends to result in under-segmentation, I tune the parameters of Omnipose to purposefully over-segment cells. In an editor like Photoshop or Napari, it is very easy to merge several partial masks together using a fill tool. In general, over-segmented masks derived from imperfect model predictions still follow the true boundaries of cells quite well, and thus eliminate much of the manual boundary tracing required to split under-segmented cells or draw cell masks from scratch.

A new model can be trained after each new addition to the training set is finished. The predictions of each model should form the basis of the next ground-truth image annotations. This should repeat until no significant edits need to be made to the model predictions.

6.6 Time lapse annotation

Much as correcting 2D predictions is easier and generally more accurate than drawing them from scratch, starting with an automatic frame-by-frame segmentation of a time lapse makes an impossible task bearable (or at least, less sanity-withering). Automatic cell tracking algorithms such as TrackMate do a great job at linking frame predictions into spacetime labels as long as cells do not move too much or have any segmentation errors [61]. However, this automatic cell linking inevitably fails where the initial segmentation has any errors, like mother-daughter merging, and so I typically find it more efficient to link cells by hand.

Manual cell linking is surprisingly easy. Loading segmentation stacks into Napari, I simply use a fill tool to recolor all slices of a cell's spacetime volume with a unique color (or, as noted, one of 6-7 colors). As with 2D, it is better to start with over-segmented labels because these are effortlessly merged with the fill tool. Splitting under-segmented cells remains a bottleneck.

Chapter 7

MISCELLANEOUS METHODS

Historically, phase contrast microscopy has been the modality of choice for studying yeast and bacteria (among other microorganisms) because cells appear dark against a light background. This contrast makes it both straightforward to find and focus on cells in a planar sample but also makes it possible to apply traditional image processing techniques for crude cell segmentation. Although microscope calibration is beyond the scope of this document, I will speak to the methods used to capture and process the images presented in the previous chapters.

7.1 Growth, density, and substrate

Image acquisition integrates photon count per-pixel over some finite period of time. To avoid motion blur, cells must be immobilized on a substrate. A thin ($\sim 0.5\,\text{mm}$ thick) pad of 1–8% agarose gel made with a nutrient media provides an ideal single-plane environment on which to deposit cells. Multiple drops of $\sim 0.5\,\mu\text{L}$ sample (cells suspended in media) can be placed on the pad and allowed to dry. This pad is mounted between a glass slide and a cover slip, sealed with got glue, and imaged in a heated microscope chamber.

Agarose pads are made by mixing and melting a liquid media with agarose powder and any desired additives. Small volumes can be made easily in a flask and microwave. This mixture should be very hot and free of bubbles when poured into a mold of uniform height. I find that glass slides separated by two layers of tape works well to give very flat, thin pads. Flatness of the pad helps to ensure that the entire field of view remains in focus (assuming the microscope mount is level). Thinness of the pad helps to reduce the scattered light and enhance contrast. Given that bacteria cells are on the order $1\,\mu\text{m}$ in width, we can safely

assume that pads $\sim 100\times$ thicker will provide a sufficient reservoir of nutrient to support cell growth for the duration of most experiments.

Because resolution in the optical axis of any microscopy technique is roughly $2\times$ worse than in the imaging plane, it is generally not possible to segment bacterial cells in 3D based on z-stacked phase contrast or epifluorescence images. We therefore require that cells grow in a single layer within the focal plane, allowing for robust 2D segmentation and tracking. 2D imaging also allows for a single image acquisition per-channel per-timepoint, reducing phototoxicity and bleaching of fluorophores. I find that 3% agarose works well to inhibit cells from growing into the pad or on top of each other for several (> 8) cell divisions.

Cell density in the deposited sample varies by application. As a droplet dries, the cell density at the periphery becomes quite high (colloquially referred to as a coffee stain). The middle of the droplet is lower in density. Roughly $\mathrm{OD}_{600} = 0.1$–0.5 provides enough cell separation for short time lapses. Overnight cultures at about $\mathrm{OD}_{600} = 1$ can form completely cell-packed fields of view that are largely unilaminar. For long time lapses, especially of mutants that grow large, we target much lower cell density. Low cell density also allows for image crops with no partial cells at the crop perimeter, which is ideal for making ground truth annotations.

7.2 Phase contrast and fluorescence microscopy

In-house imaging was performed on a Nikon Eclipse Ti-E wide-field epi-fluorescence microscope, equipped with a sCMOS camera (Hamamatsu) and X-cite LED for fluorescence imaging. We imaged through 60X and 100X 1.4 NA oil-immersion PH3 objectives. The microscope was controlled by NIS-Elements v3.30.02. Cell samples were spotted on a 3% (w/v) agarose pad placed on a microscope slide. The microscope chamber was heated to 30C or 37C when needed for time-lapse experiments.

Several images in our dataset were taken by two other laboratories using three distinct microscope/camera configurations. The Brun lab provided images of *C. crescentus* acquired on a Nikon Ti-E microscope equipped with a Photometrics Prime 95B sCMOS camera. Images were captured through a 60X Plan Apo λ 100X 1.45 NA oil Ph3 DM objective. The Wiggins lab provided *E. coli* and *A. baylyi* time lapses from both a Nikon Ti-E microscope using NIS-Elements v4.10.01 as well as a custom-built tabletop microscope using Micro-Manager v1.4, both described in previous studies [60, 65].

7.3 Exposure and outliers

Raw data is often under- or over-exposed and can contain outliers where pixels are saturated. These outliers can be small pieces of debris (especially glass) or occluded/dead sensor pixels. Phase contrast or other transmissive modalities like DIC or brightfield can be easily calibrated such that all pixels fall within the minimum and maximum sensitivity range, or gamut, of the sensor. This can be done by adjusting the illumination intensity, the exposure time, or the gain of the sensor itself (the former two are preferred to reduce noise). Visualizations like the LUT (look-up-table) histogram display the distribution of pixel intensities and should be used to calibrate exposure prior to image acquisition. Fluorescence images, on the other hand, typically cannot be adjusted to use the full gamut of the sensor because the signal is too dim.

Microscope images are typically saved in a 16-bit format, meaning that pixels may take values in the range 0 to 65535 ($2^{16} - 1$).* Pixels in 8-bit format, more commonly used in publication images, take values in the range 0 to 255 ($2^8 - 1$). When images are too dark or too bright, the pixels only occupy the bottom or top several bits, resulting in a limited number of distinct values. These "posterized" images not only look bad, but are of limited scientific value due to the loss information.

*In some software that save directly to TIF, 2 bits are reserved for metadata, so images may only be 14 bits for a range of 0–16, 383. Proprietary formats like ND2 save full 16 bit images and store metadata separately.

In practice, images should be slightly under-exposed to account for bright outliers and modality artifacts (like bright halos in phase contrast) that vary from FoV to FoV. For display and image processing, raw images are cast to the float data type (typically 32- or 64-bit) and normalized to that pixels take on values between 0 and 1. Typically, this is done via min-max scaling, where the minimum value in the image is set to 0 and the maximum value is set to 1: $I' = \frac{I - \min(I)}{\max(I) - \min(I)}$.

However, outliers (dark or light pixels well outside the typical range of the image) define the minimum and maximum of the image, and so the rescaled image compresses the signal into a smaller range than desired (Fig. 7.1). Low-contrast regions (such as deep within a bacterial microcolony) depend on the preservation of subtle numerical differences between adjacent pixels to allow for image segmentation, and this information is lost when such a compressed image is saved in a standard 16- or 8-bit format.

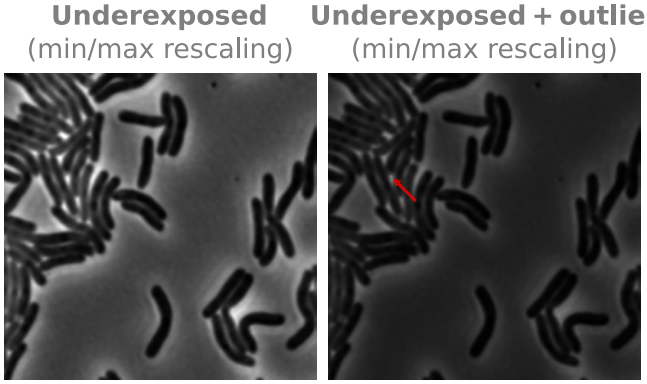

Figure 7.1: Min-max rescaling compresses signal and leads to information loss. Left: min-max rescaling of a slightly underexposed exposed image with no outliers. Right: min-max rescaling of a slightly underexposed exposed image with one outlier. This is simulated by dividing the rescaled image by 2 and adding setting one pixel to a value of 1.

The solution to outlier-induced compression is to rescale based on extremal percentiles rather than the absolute minimum and maximum of a given image. Typically, taking the 0.01st percentile as the minimum and the 99.99th percentile as the maximum will effectively ignore pixel-level outliers in both dense and sparsely populated FoVs (Fig. 7.2).

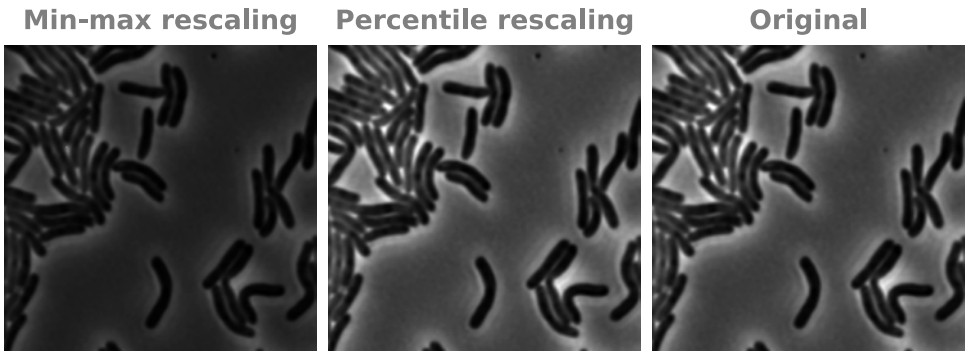

Figure 7.2: Percentile rescaling resolves outlier-induced information loss. Left: min-max rescaling with outlier pixel. Middle: percentile rescaling with outlier pixel. Right: original image (min-max rescaled).

7.4 Gamma adjustment

Images that have been successfully rescaled such that their pixels are in the range 0 to 1 can be further adjusted to reveal details in dark or light regions of the image. We may raise each pixel of an image $I \in (0, 1)$ to some power $x \in (0, \infty)$ without changing the overall range of values in the image, as $0^x = 0$ and $1^x = 1$ (Fig. 7.3).

7.5 Semantic gamma normalization

We can next use image segmentation in combination with gamma adjustment to normalize image brightness. This is very handy for making figures with images coming from different microscopes or optical configurations (Fig. 7.4).* To do this, we first use our semantic segmentation to calculate the average background signal. The gamma exponent is then the ratio of the logarithms of the target value and the measured value.

*Even within a single experiment, rescaling per-frame may images to look relatively dim or bright due to changes in cell density and random noise affecting minimum/maximum brightness. This should be handled by rescaling based on percentiles of the whole time stack. In the case of photobleaching in fluorescence channels where per-frame correction is desired, one may instead rescale each frame and then gamma-normalize so that the background noise is consistent. This function is in `omnipose.utils.normalize_stack()`

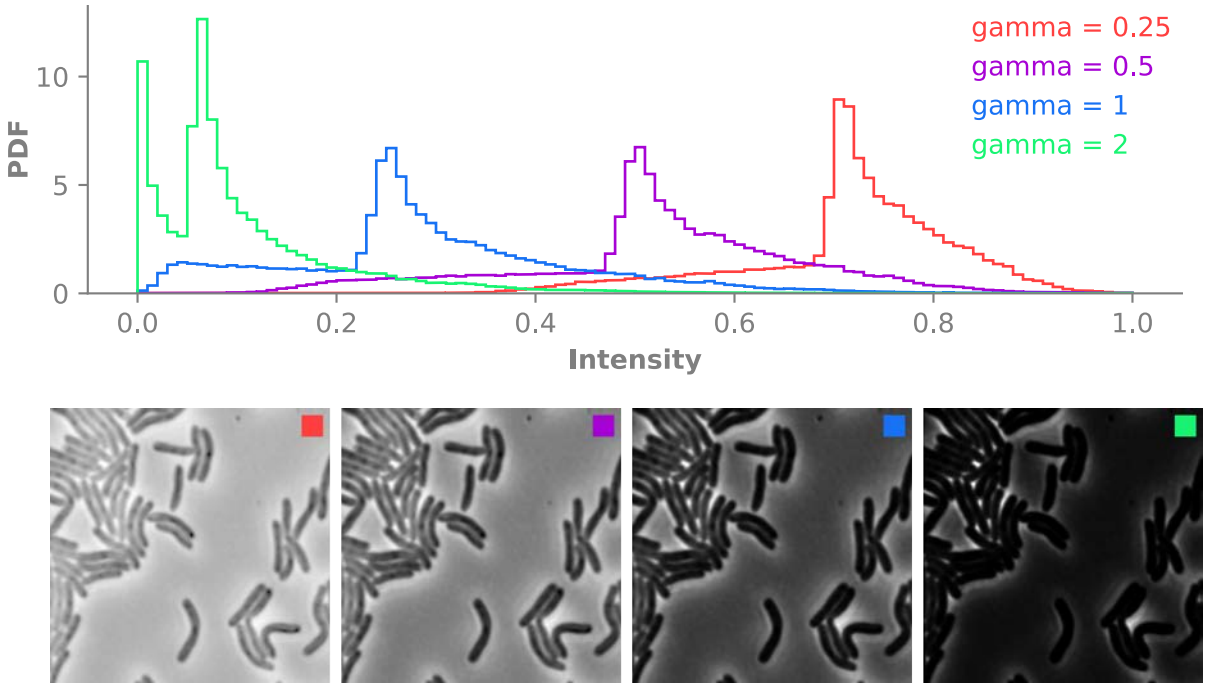

Figure 7.3: Gamma adjustment applied to a rescaled 0-1 image. Top: histogram of pixel intensities for four gamma exponentiations. Bottom: images with gamma adjustment.

7.6 Dataset processing

In the development of Omnipose, several external datasets were collected and processed (Table 2.1) [56]. Some datasets required significant pruning and/or cleanup to be useful for our study.

7.6.1 C. elegans data preparation

We obtained a 1000-frame time lapse of *C. elegans* from the Wormpose [51] GitHub adapted from the Open Worm Movement database [52], which is inaccessible at the time of writing. We also utilized the Broad Bioimage Benchmarking Collection set BBBC01040, a set of 100 images containing live and dead *C. elegans*. These images were manually cropped to select regions of each image without *C. elegans* overlaps. For both of these datasets, images were initially segmented with Omnipose to select foreground, automatically cropped to

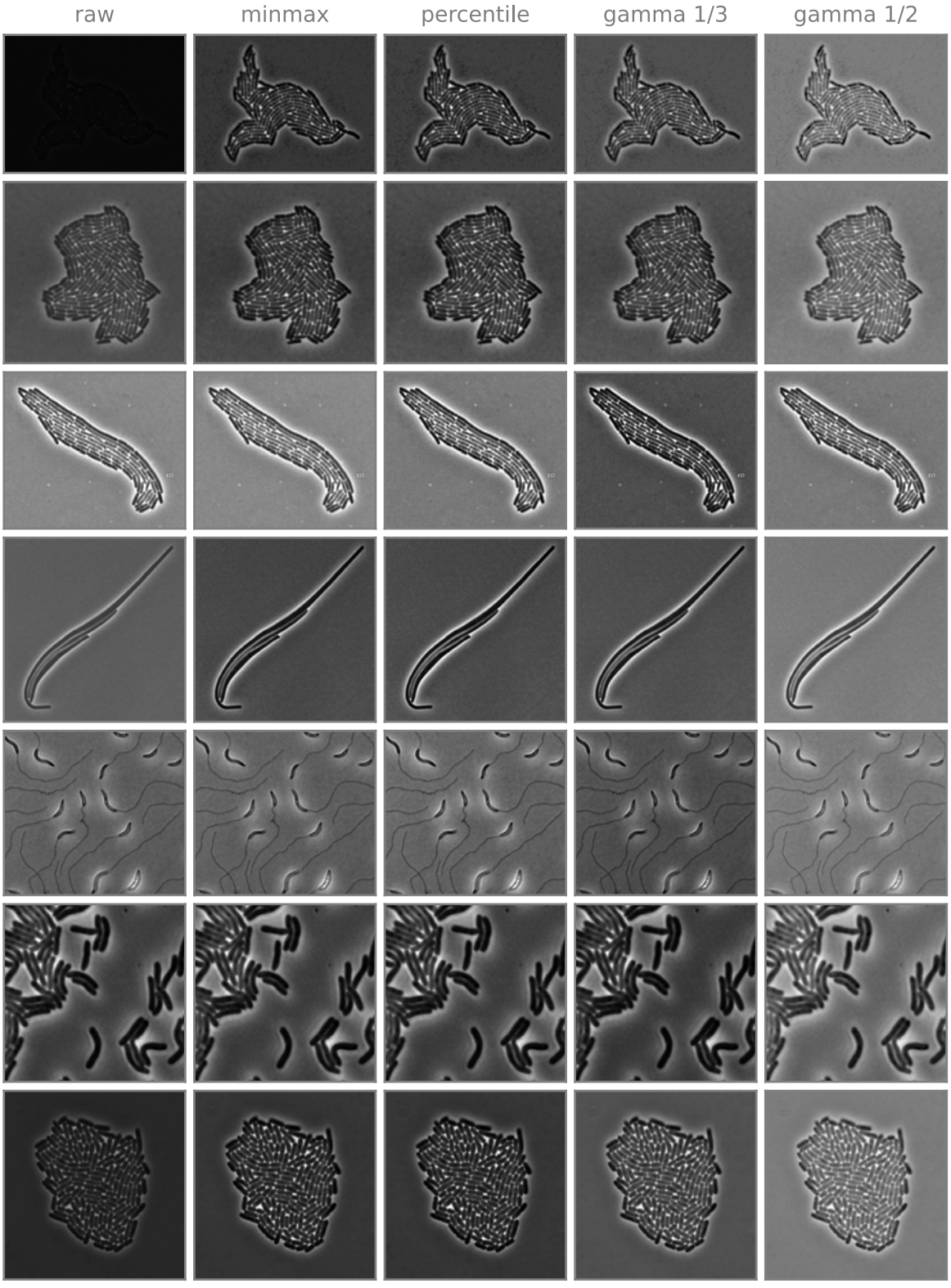

Figure 7.4: Semantic gamma normalization in comparison to no rescaling (raw), minmax, percentile, and semantic gamma normalization with two target background values (1/3, 1/2).

select individual *C. elegans* or clusters of *C. elegans*, and then packed into ensemble images for efficient annotation, training, and testing following the same procedures described below for our bacterial datasets.

For high-resolution *C. elegans* images, one gravid non-starved NGM plate of WT *C. elegans* (QZ0) was resuspended in a 1 mL M9 defined buffer. The worm suspension was pelted by quick spinning and resuspended in 100 µL of fresh M9 buffer. 20 ul of the pellet was placed on agar pads (0.3% agar, SeaKem) mounted on regular microscope slides (25 mm x 75 mm). The 20ul drops were left to dry by approx 50% of their volume at room temperature, allowing worms to arrange longitudinally, before a glass coverslip (22 mm x 22 mm) is placed on top. The sample was imaged using a Andor Dragonfly Spinning disk confocal mounted on a Nikon TiE2 microscope at 15x magnification (10x/).25 NA Nikon objective and 1.5x camera magnification) in the Brightfield channel. For each field of view, a Z stack of 7 frames centered on the worm focal plane and spanning over approx. 85 um (12.14 um spacing between frames) was acquired.

7.6.2 Arabidopsis thaliana data preparation

This specific subset of the PlantSeg dataset [19] was chosen because it represented diverse morphologies and because other subsets of their published ground truth were not in accessible formats. Three folders were provided: test, train, and val. We first combined the test and val volumes into a single test dataset. The PlantSeg algorithm appears to exclude regions labelled by 0 as during training, whereas 1 denotes background. This is incompatible with Omnipose and most other algorithm training pipelines. We therefore discarded images with the label 0 from our training set and subtracted 1 so that the remaining images conform to the convention of 0 for background. The final training set consists of Movie1 timepoints 3, 9, 35, 40, 45, and 49. The published test dataset consists of Movie2 frames 10 and 20 and Movie1 frames 4, 6, 30, and 45. Frame 45 is mistakenly duplicated from the training set,

so we discarded it from our test metrics. Only frame 30 of Movie 1 contained no exclusion zones (label 0), so we present both the performance on this subset alone as well as the performance on the full test dataset (Movie1 4, 6, 30 and Movie2 10, 20) (Fig. 3.5e). For the purposes of computing performance, exclusion zones are treated as background, and therefore all predicted labels in these regions significantly decrease the Jaccard Index for all algorithms at all IoU thresholds regardless of segmentation accuracy elsewhere. The images and ground truth were down-sampled by a factor of 3 to allow full cell cross-sections to be loaded onto the GPU during training (see below section on ND), with linear interpolation on images and nearest neighbor interpolation on the ground-truth masks.

7.6.3 Bacterial sample preparation

To image antibiotic-induced phenotypes, cells were grown without antibiotics overnight in LB, back-diluted, and spotted on agarose pads with $50\,\mu g\,mL^{-1}$ A22 or $10\,\mu g\,mL^{-1}$ cephalexin. Time lapses were captured of *E. coli* DH5α and *S. flexneri* M90T growing on these pads. *E. coli* CS703-1 was back-diluted into LB containing $1\,\mu g\,mL^{-1}$ aztreonam and spotted onto a pad without antibiotics [36]. Cells constitutively expressed GFP to visualize cell boundaries.

H. pylori LSH100 grown with and without aztreonam was provided by the Salama lab [39, 66]. Samples were fixed and stained with Alexaflour 488 to visualize the cell membrane. Images were taken on LB pads. The typical technique of allowing the spot to dry on the pad caused cells to curl up on themselves, so our images were taken by placing the cover slip on the pad immediately after spotting and applying pressure to force out excess media.

C. crescentus was cultivated and imaged by the Brun lab [40, 67]. Cells were grown in PYE, washed twice in water prior to 1:20 dilution in Hutner base-imidazole-buffered-glucose-glutamate (HIGG media) and grown at 26C for 72h. Cells were spotted on a 1% agarose PYE pads prior to imaging.

S. pristinaespiralis NRRL 2958 was grown using the following media recipe: Yeast extract 4g/L, Malt extract 10g/L, Dextrose 4g/L, Agar 20g/L. This media was used to first culture the bacteria in liquid overnight and then on a pad under the microscope. This strain forms aggregates in liquid media, so these aggregates were allowed to grow for several hours on a slide in the heated microscope chamber until we could see individual filaments extending from the aggregates. Fields of view were selected and cropped to exclude cell overlaps. Autofluorescence was captured to aid in manual segmentation.

Mixtures of *S. proteamaculans* attTn7::Km-gfp *tre1* or *tre1*^{E415Q} and *E. coli* were spotted on a PBS pad to prevent further growth. Phase-contrast images of the cells were acquired before and after a 20hr competition on a high-salt LB plate. Fluorescence images in the GFP channel were also acquired to distinguish *S. proteamaculans* from unlabeled *E. coli*.

All other individual strains in Table 2.1 were grown overnight, diluted 1:100 into fresh LB media, and grown for 1-3 hours before imaging. Mixtures were made by combining back-diluted cells roughly 1:1 by OD_{600}.

7.7 Defining the Omnipose prediction classes

Omnipose predicts four classes: two flow components, the distance field, and a boundary field. Our distance field is found by solving the eikonal equation

$$|\vec{\nabla}\phi(\vec{x})| = \frac{1}{f(\vec{x})}$$

where f represents the speed at a point $\vec{x}$. The Godunov upwind discretization of the eikonal equation is

$$\left(\frac{\max\left(\phi_{i,j} - \min\left(\phi_{i-1,j}, \phi_{i+1,j}\right), 0\right)}{\Delta x}\right)^2 + \left(\frac{\max\left(\phi_{i,j} - \min\left(\phi_{1,j-1}, \phi_{i,j+1}\right), 0\right)}{\Delta y}\right)^2 = \frac{1}{f_{i,j}}$$

Our solution to this equation is based on the Improved FIM Algorithm 1.1 of Huang [47], as follows. Our key contribution to this algorithm is the addition of ordinal sampling to boost both convergence and smoothness of the final distance field.

2D update function for $\phi_{i,j}$ on a cartesian grid

1. Find neighboring points for cardinal axes ($\Delta x = \Delta y = \delta$) :

$$\phi^{\min x} = \min\left(\phi_{i-1,j}, \phi_{i+1,j}\right), \quad \phi^{\text{miny}} = \min\left(\phi_{i,j-1}, \phi_{i,j+1}\right)$$

2. Find neighboring points for ordinal axes $\left(\hat{x} \cdot \hat{a} = \hat{y} \cdot \hat{b} = \frac{\sqrt{2}}{2}, \frac{\Delta a}{\Delta x} = \frac{\Delta b}{\Delta y} = \sqrt{2}\delta\right)$:

$$\phi^{\text{mina}} = \min\left(\phi_{i-1,j-1}, \phi_{i+1,j+1}\right), \quad \phi^{\text{minb}} = \min\left(\phi_{i+1,j-1}, \phi_{i-1,j+1}\right)$$

3. Calculate update along cardinal axes:

   if $\left|\phi^{\text{minx}} - \phi^{\text{miny}}\right| > \frac{\sqrt{2}\delta}{f_{i,j}}$ **then**

   $U^{xy} = \min\left(\phi^{\text{minx}}, \phi^{\text{miny}}\right) + \frac{\delta}{f_{i,j}}$

   else

   $U^{xy} = \frac{1}{2}\left(\phi^{\min x} + \phi^{\text{miny}} + \sqrt{2\left(\frac{\delta}{f_{i,j}}\right)^2 - \left(\phi^{\text{minx}} - \phi^{\text{miny}}\right)^2}\right)$

4. Calculate update along ordinal axes:

   if $\left|\phi^{\text{mina}} - \phi^{\text{minb}}\right| > \frac{2\delta}{f_{i,j}}$ **then**

   $U^{ab} = \min\left(\phi^{\text{mina}}, \phi^{\text{minb}}\right) + \frac{\sqrt{2}\delta}{f_{i,j}}$

   else

   $U^{ab} = \frac{1}{2}\left(\phi^{\text{mina}} + \phi^{\text{minb}} + \sqrt{4\left(\frac{\delta}{f_{i,j}}\right)^2 - \left(\phi^{\text{mina}} - \phi^{\text{minb}}\right)^2}\right)$

5. Update with geometric mean:

$$\phi_{i,j} = \sqrt{U^{xy} U^{ab}}$$

This update rule is repeated until convergence (Fig. 3.1). We take $\delta = f_{i,j}$ to obtain the signed distance field used in Omnipose. The flow field components are defined by the normalized gradient of this distance field ϕ. The boundary field is defined by points satisfying $0 < \phi < 1$. For network prediction, the boundary map is converted to the logits (inverse sigmoid) representation, such that points in the range $[0, 1]$ are mapped to $[-5, 5]$. For consistent value ranges across prediction classes, the flow components are multiplied by 5 and all background values of the distance field ($\phi = 0$) are set to -5.

Chapter 8

OUTLOOK

In this document, I have argued that pixel-level accuracy is required for image cytometry of single-cell organisms. Although I have shown Omnipose to be the current state-of-the-art in achieving this goal, there are several facets of the image segmentation problem where more testing and future research is required.

8.1 The future of image segmentation algorithms

My results show definitively that DNN approaches for image segmentation are capable of solving virtually any segmentation task that a human can perform. However, it is unknown whether the U-net architecture, or indeed DNNs in general, are the best choice for performing the many-to-one image transformations that we seek.[†] I can imagine that we will one day have an alternative universal function approximator for Omnipose, one that does not require as much training data, that trains faster, and that evaluates faster. However, the U-net appears to be sufficient for all our needs so far, and I have consequently spent little time working on it besides generalizing the Cellpose implementation to 3D.

Regardless of model architecture, the energy landscape defined by the loss function determines the ultimate performance of the DNN. Therefore, loss functions represent an opportunity for improving training time and prediction quality. Over the development of Omnipose, I designed several custom loss functions and found that a carefully weighted sum of each was critical to achieving the performance we ultimately obtained. In the future, I foresee that adversarial loss functions (loss functions that are themselves learned) will either replace or augment manually defined loss functions.

[†]DNNs are generally not bijective functions. In the case of all the approaches outlined here, images of the same cells under different modalities would correspond to the same DNN output.

Hyperparameters, such as the weighting between loss terms, learning rate, decay rate, dropout, etc., play an important role in defining and navigating the energy landscape. Manually tuning these is an art, but a robust grid search would be closer to a science.* I suspect that this exercise could give us slightly better models and perhaps shorter training times and more stable convergence.

Obtaining an accurate (enough) approximation to the desired function is only the first step of image segmentation with a DNN. The second is mask reconstruction. As noted in Chapter 4, any post-processing of DNN output should include error correction. Euler integration works well for Omnipose, but any error correction scheme must be tailored to the prediction classes of the DNN. Likewise, the mask reconstruction algorithm is always designed specifically for the prediction classes. Future research should consider alternatives to the distance and flow field outputs of Omnipose, but careful thought must be put towards both error correction and mask reconstruction schemes.

Finally, we may consider how image segmentation is encoded. The affinity graph representing connections between adjacent hypervoxels appears to be sufficient for any segmentation task of contiguous hypervolumes. However, alternative approaches may be required for discontiguous hypervolumes.

8.1.1 Low-frame-rate time lapses

If cells move so far as to have little to know overlap between frames, we will not be able to segment them as a contiguous object in 3D. However, it is possible to still obtain tracking

*Understanding and developing a theoretical framework for the influence of each would be much better, but I have my doubts about the feasibility of this.

information if we allow an affinity graph to have edges between non-adjacent pixels. An efficient implementation of this may be worth some research.*

8.1.2 Cell overlap

Cell overlap in 2D imaging is another niche problem that might be solved by an alternative segmentation encoding. For example, we could output two instance masks – one for foreground, one for background (or however many layers are needed). This could be useful for properly handling data with common and unavoidable overlap, such as tissue slices. Although overlapping signal would need to be excluded from analysis, this multi-plane segmentation would at least allow for accurate cell counts and morphology to be extracted from such images.

8.1.3 Hyperspectral imaging

Fluorescent molecules generally have multiple emission peaks, and so cells imaged in multiple channels may appear in a discontiguous subset of channels. Again, an affinity graph with edges between nonadjacent nodes would allow the data in multiple uncorrelated channels to be detected as part of a single cell.

8.2 The future of ground truth data

No matter the implementation, example data will continue to be a critical component of any future segmentation algorithm. My ground truth data took hundreds of hours to collect and annotate, and despite expertise in the field, I cannot be sure of its physiological accuracy. Moreover, this data is largely irrelevant to other kinds of images not represented by my data.

---

*However, my advice would be that this is not a problem worth solving. As Paul says: fix the data, not the analysis. Unless you have some crazy photosensitive bugs, you can take as many phase / brightfield images as you need to do cell tracking and collect fluorescence less frequently if needed.

In order to create equally precise models on different cell types and imaging modalities, we require a more scalable method of generating training data.

To do this, we need to simulate both images and ground truth cell labels. Both are an active area of research. Images can, in principle, be simulated with sufficient compute and with high enough detail in a physical optics model. The software SyMBac is a start to this [68].

Alternatively, DNNs may be used to solve this problem with less fine tuning and with greater fidelity to target image styles. Generative adversarial networks (GANs) and diffusion models are both promising approaches, but they of course require training data. Furthermore, it is an open question as to whether diffusion models can be constrained to output signals that correspond to ground truth mask boundaries with sub-pixel-level precision.

Simulation of cell growth as well as the internal structure that causes the high-frequency variations in signal within cells is an even more challenging problem. Ball-and-spring models are a promising, tractable approach for cell growth simulation, but current implementations are crude and slow. Moreover, such implementations do not accurately model cell septum formation or internal structure.

8.3 The future of Omnipose

With these points in mind, I conclude that there are two components of Omnipose that are unlikely to obsolesce in the near future.

1. The network predictions

   Distance field and its gradient, the flow field, are highly efficient compared to direct affinity graph prediction. The former requires just $(1 + N)$ prediction classes in N dimensions, but the latter would require $(3^N - 1)/2$ for adjacent hypervoxel edge predictions (more if nonadjacent hypervoxel connections are allowed). The flow field is

also highly amenable to error correction, but it is unclear how to do this efficiently for affinity graphs. Although it may be a lack of imagination on my part, I suspect that some form of morphology-independent, vector-field-based network output will remain the state-of-the-art for some time.*

2. Affinity graph reconstruction

   Euler integration is simple, effective, and efficient. My contribution to this idea was to suppress pixel movement so that cells are not over-segmented.† I think any vector-field-based method will continue to take advantage of this. Likewise, I expect that my general approach of determining affinity graphs from hypervoxel displacement will be applicable to any Euler-integrated vector field output. The primary improvement I see moving forward is implementing the last step, connected components labeling, on the GPU.

Meanwhile, I expect that advancements in DNN architecture will replace the Omnipose U-net in the next several years. More importantly, I anticipate that manual ground truth annotation of large datasets will soon be a thing of the past. Training segmentation models will require far less manual annotation, if any, as we continue to develop means of synthesizing images that represent the subtle intricacies of real data.

---

*There may be alternative vector fields to test, but I think that each would ultimately be parallel to the Omnipose flow but with nonuniform magnitude.

†Explicit suppression is no longer needed in the affinity segmentation pipeline, as we only require comparisons of displacement direction to determine connectivity (rather than requiring connected blobs along the skeleton for cluster identification). This means that we can use fewer iterations (which is faster to compute). It is still useful, however, to rescale the flow by its divergence to increase movement at the boundaries and reduce movement at the skeleton.

BIBLIOGRAPHY

[1] Elvis Awuni. Status of targeting mreb for the development of antibiotics. *Frontiers in Chemistry*, 7:884, 2020.

[2] H. Jeckel and K. Drescher. Advances and opportunities in image analysis of bacterial cells and communities. *FEMS Microbiol Rev*, 45(4), 2021.

[3] A. Bali and S. N. Singh. A review on the strategies and techniques of image segmentation, 2015.

[4] A. M. Lucas, P. V. Ryder, B. Li, B. A. Cimini, K. W. Eliceiri, and A. E. Carpenter. Open-source deep-learning software for bioimage segmentation. *Mol Biol Cell*, 32(9):823–829, 2021.

[5] Jannis Ahlers, Daniel Althviz Mor, Oren Amsalem, Ashley Anderson, Grzegorz Bokota, Peter Boone, Jordo Bragantini, Genevieve Buckley, Alister Burt, Matthias Bussonnier, Ahmet Can Solak, Clment Caporal, Draga Doncila Pop, Kira Evans, Jeremy Freeman, Lorenzo Gaifas, Christoph Gohlke, Kabilar Gunalan, Hagai Har-Gil, Mark Harfouche, Kyle I. S. Harrington, Volker Hilsenstein, Katherine Hutchings, Talley Lambert, Jessy Lauer, Gregor Lichtner, Ziyang Liu, Lucy Liu, Alan Lowe, Luca Marconato, Sean Martin, Abigail McGovern, Lukasz Migas, Nadalyn Miller, Hector Muoz, Jan-Hendrik Müller, Christopher Nauroth-Kre, Juan Nunez-Iglesias, Constantin Pape, Kim Pevey, Gonzalo Pea-Castellanos, Andrea Pierr, Jaime Rodrguez-Guerra, David Ross, Loic Royer, Craig T. Russell, Gabriel Selzer, Paul Smith, Peter Sobolewski, Konstantin Sofiiuk, Nicholas Sofroniew, David Stansby, Andrew Sweet, Wouter-Michiel Vierdag, Pam Wadhwa, Melissa Weber Mendonça, Jonas Windhager, Philip Winston, and Kevin Yamauchi. napari: a multi-dimensional image viewer for python, 2023.

[6] A. Paintdakhi, B. Parry, M. Campos, I. Irnov, J. Elf, I. Surovtsev, and C. Jacobs-Wagner. Oufti: an integrated software package for high-accuracy, high-throughput quantitative microscopy analysis. *Mol Microbiol*, 99(4):767–77, 2016.

[7] A. Ducret, E. M. Quardokus, and Y. V. Brun. Microbej, a tool for high throughput bacterial cell detection and quantitative analysis. *Nat Microbiol*, 1(7):16077, 2016.

[8] A. D. Balomenos, P. Tsakanikas, Z. Aspridou, A. P. Tampakaki, K. P. Koutsoumanis, and E. S. Manolakos. Image analysis driven single-cell analytics for systems microbiology. *BMC Syst Biol*, 11(1):43, 2017.

[9] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nat Methods*, 18(1):100–106, 2021.

[10] S. Stylianidou, C. Brennan, S. B. Nissen, N. J. Kuwada, and P. A. Wiggins. Supersegger: robust image segmentation, analysis and lineage tracking of bacterial cells. *Mol Microbiol*, 102(4):690–700, 2016.

[11] R. van Raaphorst, M. Kjos, and J. W. Veening. Bactmap: An r package for integrating, analyzing and visualizing bacterial microscopy data. *Mol Microbiol*, 113(1):297–308, 2020.

[12] S. Panigrahi, D. Murat, A. Le Gall, E. Martineau, K. Goldlust, J. B. Fiche, S. Rombouts, M. Nollmann, L. Espinosa, and T. Mignot. Misic, a general deep learning-based method for the high-throughput cell segmentation of complex bacterial communities. *Elife*, 10, 2021.

[13] D. Bannon, E. Moen, M. Schwartz, E. Borba, T. Kudo, N. Greenwald, V. Vijayakumar, B. Chang, E. Pao, E. Osterman, W. Graf, and D. Van Valen. Deepcell kiosk: scaling deep learning-enabled cellular image analysis with kubernetes. *Nat Methods*, 18(1):43–45, 2021.

[14] J. B. Lugagne, H. Lin, and M. J. Dunlop. Delta: Automated cell segmentation, tracking, and lineage reconstruction using deep learning. *PLoS Comput Biol*, 16(4):e1007673, 2020.

[15] J. H. Smit, Y. Li, E. M. Warszawik, A. Herrmann, and T. Cordes. Colicoords: A python package for the analysis of bacterial fluorescence microscopy data. *PLoS One*, 14(6):e0217524, 2019.

[16] E. Czech, B. A. Aksoy, P. Aksoy, and J. Hammerbacher. Cytokit: a single-cell analysis toolkit for high dimensional fluorescent microscopy imaging. *BMC Bioinformatics*, 20(1):448, 2019.

[17] C. McQuin, A. Goodman, V. Chernyshev, L. Kamentsky, B. A. Cimini, K. W. Karhohs, M. Doan, L. Ding, S. M. Rafelski, D. Thirstrup, W. Wiegraebe, S. Singh, T. Becker, J. C. Caicedo, and A. E. Carpenter. Cellprofiler 3.0: Next-generation image processing for biology. *PLoS Biol*, 16(7):e2005970, 2018.

[18] S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, K. Eren, J. I. Cervantes, B. Xu, F. Beuttenmueller, A. Wolny, C. Zhang, U. Koethe, F. A. Hamprecht, and A. Kreshuk. ilastik: interactive machine learning for (bio)image analysis. *Nat Methods*, 16(12):1226–1232, 2019.

[19] A. Wolny, L. Cerrone, A. Vijayan, R. Tofanelli, A. V. Barro, M. Louveaux, C. Wenzl, S. Strauss, D. Wilson-Sanchez, R. Lymbouridou, S. S. Steigleder, C. Pape, A. Bailoni, S. Duran-Nebreda, G. W. Bassel, J. U. Lohmann, M. Tsiantis, F. A. Hamprecht, K. Schneitz, A. Maizel, and A. Kreshuk. Accurate and versatile 3d segmentation of plant tissues at cellular resolution. *Elife*, 9, 2020.

[20] T. Ursell, T. K. Lee, D. Shiomi, H. Shi, C. Tropini, R. D. Monds, A. Colavin, G. Billings, I. Bhaya-Grossman, M. Broxton, B. E. Huang, H. Niki, and K. C. Huang. Rapid, precise quantification of bacterial cellular dimensions across a genomic-scale knockout library. *BMC Biol*, 15(1):17, 2017.

[21] H. F. Tsai, J. Gajda, T. F. W. Sloan, A. Rares, and A. Shen. Usiigaci: Instance-aware cell tracking in stain-free phase contrast microscopy enabled by machine learning. *SoftwareX*, 9:230–237, 2019.

[22] J. Reiner, G. Azran, and G. Hyams. Microanalyzer: A python tool for automated bacterial analysis with fluorescence microscopy. *arXiv*, 2020.

[23] U. Schmidt, M. Weigert, G. Myers, A. F. Frangi, C. Davatzikos, C. Alberola-Lopez, and G. Fichtinger. *Cell Detection with Star-Convex Polygons*. Medical Image Computing and Computer Assisted Intervention – MICCAI 2018. 2018.

[24] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. *arXiv*, 2018.

[25] K. Shal and M. S. Choudhry. Evolution of deep learning algorithms for mri-based brain tumor image segmentation. *Crit Rev Biomed Eng*, 49(1):77–94, 2021.

[26] D. T. Kysela, A. M. Randich, P. D. Caccamo, and Y. V. Brun. Diversity takes shape: Understanding the mechanistic and adaptive basis of bacterial morphology. *PLoS Biol*, 14(10):e1002565, 2016.

[27] S. E. Jones and M. A. Elliot. 'exploring' the regulation of streptomyces growth and development. *Curr Opin Microbiol*, 42:25–30, 2018.

[28] P. D. Caccamo and Y. V. Brun. The molecular basis of noncanonical bacterial morphology. *Trends Microbiol*, 26(3):191–208, 2018.

[29] B. Behera, G. K. Anil Vishnu, S. Chatterjee, V. Vsn Sitaramgupta, N. Sreekumar, A. Nagabhushan, N. Rajendran, B. H. Prathik, and H. J. Pandya. Emerging technologies for antibiotic susceptibility testing. *Biosens Bioelectron*, 142:111552, 2019.

[30] C. Tropini, K. A. Earle, K. C. Huang, and J. L. Sonnenburg. The gut microbiome: Connecting spatial organization to function. *Cell Host Microbe*, 21(4):433–442, 2017.

[31] R. Hartmann, M. C. F. van Teeseling, M. Thanbichler, and K. Drescher. Bacstalk: A comprehensive and interactive image analysis software tool for bacterial cell biology. *Mol Microbiol*, 114(1):140–150, 2020.

[32] A. Goni-Moreno, J. Kim, and V. de Lorenzo. Cellshape: A user-friendly image analysis tool for quantitative visualization of bacterial cell factories inside. *Biotechnol J*, 12(2), 2017.

[33] J. W. Young, J. C. Locke, A. Altinok, N. Rosenfeld, T. Bacarian, P. S. Swain, E. Mjolsness, and M. B. Elowitz. Measuring single-cell gene expression dynamics in bacteria using fluorescence time-lapse microscopy. *Nat Protoc*, 7(1):80–8, 2011.

[34] O. M. OConnor, R. N. Alnahhas, J. B. Lugagne, and M. J. Dunlop. Delta 2.0: A deep learning pipeline for quantifying single-cell spatial and temporal dynamics. *bioRxiv*, 2021.

[35] G. J. Bean, S. T. Flickinger, W. M. Westler, M. E. McCully, D. Sept, D. B. Weibel, and K. J. Amann. A22 disrupts the bacterial actin cytoskeleton by directly binding and inducing a low-affinity state in mreb. *Biochemistry*, 48(22):4852–7, 2009.

[36] B. M. Meberg, F. C. Sailer, D. E. Nelson, and K. D. Young. Reconstruction of escherichia coli mrca (pbp 1a) mutants lacking multiple combinations of penicillin binding proteins. *J Bacteriol*, 183(20):6148–9, 2001.

[37] Valérie Barbe, David Vallenet, Nuria Fonknechten, Annett Kreimeyer, Sophie Oztas, Laurent Labarre, Stéphane Cruveiller, Catherine Robert, Simone Duprat, Patrick Wincker, L Nicholas Ornston, Jean Weissenbach, Philippe Marlière, Georges N Cohen, and Claudine Médigue. Unique features revealed by the genome sequence of acinetobacter sp. adp1, a versatile and naturally transformation competent bacterium. *Nucleic Acids Res*, 32(19):5766–5779, 2004.

[38] Yiting Yu, H Stanley Kim, Hui Hoon Chua, Chi Ho Lin, Siew Hoon Sim, Daoxun Lin, Alan Derr, Reinhard Engels, David DeShazer, Bruce Birren, William C Nierman, and Patrick Tan. Genomic patterns of pathogen evolution revealed by comparison of burkholderia pseudomallei, the causative agent of melioidosis, to avirulent burkholderia thailandensis. *BMC Microbiol*, 6:46, May 2006.

[39] A. C. Lowenthal, M. Hill, L. K. Sycuro, K. Mehmood, N. R. Salama, and K. M. Ottemann. Functional analysis of the helicobacter pylori flagellar switch proteins. *J Bacteriol*, 191(23):7147–56, 2009.

[40] M. Evinger and N. Agabian. Envelope-associated nucleoid from caulobacter crescentus stalked and swarmer cells. *J Bacteriol*, 132(1):294–301, 1977.

[41] Anna Allué-Guardia, Mylea Echazarreta, Sara S K Koenig, Karl E Klose, and Mark Eppinger. Closed genome sequence of vibrio cholerae o1 el tor inaba strain a1552. *Genome Announc*, 6(9), Mar 2018.

[42] C. K. Stover, X. Q. Pham, A. L. Erwin, S. D. Mizoguchi, P. Warrener, M. J. Hickey, F. S. L. Brinkman, W. O. Hufnagle, D. J. Kowalik, M. Lagrou, R. L. Garber, L. Goltry, E. Tolentino, S. Westbrock-Wadman, Y. Yuan, L. L. Brody, S. N. Coulter, K. R. Folger, A. Kas, K. Larbig, R. Lim, K. Smith, D. Spencer, G. K. S. Wong, Z. Wu, I. T. Paulsen, J. Reizer, M. H. Saier, R. E. W. Hancock, S. Lory, and M. V. Olson. Complete genome sequence of pseudomonas aeruginosa pao1, an opportunistic pathogen. *Nature*, 406(6799):959–964, 2000.

[43] R. F. Laine, I. Arganda-Carreras, R. Henriques, and G. Jacquemet. Avoiding a replication crisis in deep-learning-based bioimage analysis. *Nat Methods*, 18(10):1136–1144, 2021.

[44] A. A. Taha and A. Hanbury. Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool. *BMC Med Imaging*, 15:29, 2015.

[45] W. Lu, Y. Cheng, C. Xiao, S. Chang, S. Huang, B. Liang, and T. Huang. Unsupervised sequential outlier detection with deep architectures. *IEEE Trans Image Process*, 26(9):4321–4330, 2017.

[46] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static hamilton-jacobi equations. *Proc Natl Acad Sci U S A*, 98(20):11069–74, 2001.

[47] Y. Huang. Improved fast iterative algorithm for eikonal equation for gpu computing. *arXiv*, 2021.

[48] M. Ester, H.P. Kreigel, J. Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.

[49] Z. Gitai. New fluorescence microscopy methods for microbiology: sharper, faster, and quantitative. *Curr Opin Microbiol*, 12(3):341–6, 2009.

[50] L. R. Girard, T. J. Fiedler, T. W. Harris, F. Carvalho, I. Antoshechkin, M. Han, P. W. Sternberg, L. D. Stein, and M. Chalfie. Wormbook: the online review of caenorhabditis elegans biology. *Nucleic Acids Res*, 35(Database issue):D472–5, 2007.

[51] L. Hebert, T. Ahamed, A. C. Costa, L. O'Shaughnessy, and G. J. Stephens. Wormpose: Image synthesis and convolutional networks for pose estimation in c. elegans. *PLoS Comput Biol*, 17(4):e1008914, 2021.

[52] A. Javer, M. Currie, C. W. Lee, J. Hokanson, K. Li, C. N. Martineau, E. Yemini, L. J. Grundy, C. Li, Q. Ch'ng, W. R. Schafer, E. A. A. Nollen, R. Kerr, and A. E. X. Brown. An open-source platform for analyzing and sharing worm-behavior data. *Nat Methods*, 15(9):645–646, 2018.

[53] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter. Annotated high-throughput microscopy image sets for validation. *Nat Methods*, 9(7):637, 2012.

[54] S. Y. Ting, D. E. Bosch, S. M. Mangiameli, M. C. Radey, S. Huang, Y. J. Park, K. A. Kelly, S. K. Filip, Y. A. Goo, J. K. Eng, M. Allaire, D. Veesler, P. A. Wiggins, S. B. Peterson, and J. D. Mougous. Bifunctional immunity proteins protect bacteria against ftsz-targeting adp-ribosylating toxins. *Cell*, 175(5):1380–1392, 2018.

[55] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Comput.*, 22(2):511–538, February 2010.

[56] Kevin J. Cutler, Carsen Stringer, Teresa W. Lo, Luca Rappez, Nicholas Stroustrup, S. Brook Peterson, Paul A. Wiggins, and Joseph D. Mougous. Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation. *Nature Methods*, 19(11):1438–1448, 2022.

[57] Yanzhe Xu, Fei Gao, Teresa Wu, Kevin M. Bennett, Jennifer R. Charlton, and Suryadipto Sarkar. U-net with optimal thresholding for small blob detection in medical images. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1761–1767, 2019.

[58] Eric Jelli, Takuya Ohmura, Niklas Netter, Martin Abt, Eva Jiménez-Siebert, Konstantin Neuhaus, Daniel K H Rode, Carey D Nadell, and Knut Drescher. Single-cell segmentation in bacterial biofilms with an optimized deep learning method enables tracking of cell lineages and measurements of growth rates. *Mol Microbiol*, 119(6):659–676, Jun 2023.

[59] Kevin Floc'h, Françoise Lacroix, Pascale Servant, Yung-Sing Wong, Jean-Philippe Kleman, Dominique Bourgeois, and Joanna Timmins. Cell morphology and nucleoid dynamics in dividing deinococcus radiodurans. *Nature Communications*, 10(1):3815, 2019.

[60] J. Bailey, J. Cass, J. Gasper, N. D. Ngo, P. Wiggins, and C. Manoil. Essential gene deletions producing gigantic bacteria. *PLoS Genet*, 15(6):e1008195, 2019.

[61] Dmitry Ershov, Minh-Son Phan, Joanna W. Pylvänäinen, Stphane U. Rigaud, Laure Le Blanc, Arthur Charles-Orszag, James R. W. Conway, Romain F. Laine, Nathan H. Roy, Daria Bonazzi, Guillaume Dumnil, Guillaume Jacquemet, and Jean-Yves Tinevez. Trackmate 7: integrating state-of-the-art segmentation algorithms into tracking pipelines. *Nature Methods*, 19(7):829832, June 2022.

[62] Paula P Navarro, Andrea Vettiger, Virly Y Ananda, Paula Montero Llopis, Christoph Allolio, Thomas G Bernhardt, and Luke H Chao. Cell wall synthesis and remodelling dynamics determine division site architecture and cell shape in escherichia coli. *Nat. Microbiol.*, 7(10):1621–1634, October 2022.

[63] Kenneth Appel and Wolfgang Haken. *The Four-Color Problem*, pages 153–180. Springer New York, New York, NY, 1978.

[64] Neil Robertson, Daniel Sanders, Paul Seymour, and Robin Thomas. The four-colour theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2–44, 1997.

[65] J. A. Cass, S. Stylianidou, N. J. Kuwada, B. Traxler, and P. A. Wiggins. Probing bacterial cell biology using image cytometry. *Mol Microbiol*, 103(5):818–828, 2017.

[66] J. A. Taylor, B. P. Bratton, S. R. Sichel, K. M. Blair, H. M. Jacobs, K. E. DeMeester, E. Kuru, J. Gray, J. Biboy, M. S. VanNieuwenhze, W. Vollmer, C. L. Grimes, J. W. Shaevitz, and N. R. Salama. Distinct cytoskeletal proteins define zones of enhanced cell wall synthesis in helicobacter pylori. *Elife*, 9, 2020.

[67] P. D. Caccamo, M. Jacq, M. S. VanNieuwenhze, and Y. V. Brun. A division of labor in the recruitment and topological organization of a bacterial morphogenic complex. *Curr Biol*, 30(20):3908–3922 e4, 2020.

[68] Georgeos Hardo, Maximilian Noka, and Somenath Bakshi. Synthetic micrographs of bacteria (symbac) allows accurate segmentation of bacterial cells using deep neural networks. *BMC Biology*, 20(1), November 2022.