ETD

By: Nidhi Chopdekar

As of: May 24, 2024 8:59:32 PM
15,350 words - 62 matches - 55 sources

Similarity Index
7%

Mode: Similarity Report

paper text:

Image Analysis for Sliding Motility of Clostridium perfringens Nidhi Chopdekar

Thesis submitted to the faculty of the **Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Master of Science In Biological Sciences** Jing Chen **Stephen Melville** 2

Silke Hauf May 7, 2024 Blacksburg, Virginia Keywords: image analysis, image segmentation, sliding motility, live-cell images, cell tracking. Copyright 2024, Nidhi Chopdekar Image Analysis for Sliding Motility of Clostridium perfringens Nidhi Chopdekar ACADEMIC ABSTRACT The research investigates the sliding motility of Clostridium perfringens by employing machine learning-based image segmentation techniques and tracking to extract key quantitative characteristics of the movement of the bacteria. C. perfringens

cells maintain end-to-end connections after **cell divisions** and **form** elongated **chains** 47

that expand in a one-dimensional fashion. Cells in the elongating chains are pushed by each other to achieve a sliding movement at potentially high speeds. However, these chains are susceptible to breakage due to stress accumulation from rapid growth, which would undermine efficiency of the passive sliding motility. Utilizing AI-powered image analysis, this research aims to obtain detailed quantification of these dynamics and generate data for future mechanistic studies of the sliding motility. Results from this work highlight the effectiveness of machine learning in detecting individual cells from microscopy images. The accurately segmented cells enable enhanced tracking and detailed analysis of bacterial motility. The results generate useful quantitative data such as growth rate, velocity, and division frequency of C. perfringens. Image Analysis for Sliding Motility of Clostridium perfringens Nidhi Chopdekar GENERAL AUDIENCE ABSTRACT The research project explores the

movement of Clostridium perfringens, a bacterium often responsible for food poisoning, by using machine learning techniques to observe and analyze how each bacterial cell moves within its colony. These bacteria form long, chain-like structures that help them move more rapidly. However, these chains can break when they become too long and undergo too much stress. By applying artificial intelligence-based tools to automatically detect and track cells in time-lapse microscopy videos, the project provides useful data of how these bacteria slide, grow and divide. These data will help us understand the chain-based bacterial sliding in C. perfringens and its underlying mechanism.

Acknowledgements I would like to express my deepest **gratitude to everyone who has** 41

supported me throughout

the completion of my degree. **First and foremost, I would like to thank my** advisor, **Dr** . Jing Chen, **for** her invaluable guidance 36 **and**

continuous encouragement.

Her expertise and support have been instrumental in shaping this 21

research.

I am also grateful to my committee members, Dr . Silke Hauf **and Dr** . Stephen Melville, **for their** valuable suggestions **and** 38

guidance that have significantly contributed

to the quality **of this work. I would like to thank** 48

Dr. Stephen Melville for providing the dataset used in this research from his laboratory. This was critical to the success of my research.

TABLE OF CONTENTS Chapter 1: Background……6 1.1 55

Microbiology and C. perfringens……6 1.2 Sliding motility of C. perfringens……7 1.3 Machine Learning and Image Analysis……8 1.4 Machine Learning based Image Segmentation……14 1.5 Machine Learning based Image Processing for my Project……15 Chapter 2: Initial Research Strategies……19 2.1 Data Acquisition……19 2.1.1 Methods……19 2.1.2 Results……21 2.2 Segmentation of individual C. perfringens cells……21 2.2.1 Methods……22 2.2.2 Results……23 Chapter 3: Final Tasks and Research Strategies……24 3.1 Data Acquisition……24 3.2 Segmentation of cells……25 3.2.1 Methods……27 3.2.2 Results……30 3.3 Cell Tracking……32 3.3.1 Methods……33 3.3.2 Results……34 3.4 Extraction of key characteristics of C. perfringens sliding……35 3.4.1 Methods……35 3.4.2 Results……43 4 3.5 Discussion……49 Chapter 4: Conclusion……50 Appendix……52 References……57 CHAPTER 1: BACKGROUND

Clostridium Perfringens, a pathogenic bacterium commonly found in various environments, including decaying vegetation, marine sediment and the intestinal tract of humans and animals poses a significant health risk as

one of the most common causes of food poisoning in the United States (Centers for Disease Control and Prevention , 2018). **It** 26

is essential to comprehend the mechanisms by which C. perfringens spread within the host and induce diseases. The application of Artificial Intelligence and Machine Learning technologies provides a way to analyze the microscopy videos of these bacteria without human error and bias and extract key motility characteristics. One notable technique is image segmentation, an image processing technique that segments cells based on factors such as color and intensity. The segmentation provides the ability to locate individual cells in a group of bacterial cells and record their coordinates over time, thereby offering insight into the dynamics of bacterial sliding motility. In the following sections, a comprehensive outline of my research project focusing on image segmentation is provided in detail. The first section includes background, knowledge gaps, and the relevance of existing literature to my research, showcasing how the image segmentation methodologies I implemented in my project builds on the current literature in computer-based image processing. The second section clearly states the tasks, methodologies, and initial findings of my research. Finally, the third section includes the final methodologies employed and discusses the resulting outcomes. 1.1 Microbiology and C. perfringens Microbiology is a multidisciplinary field, encompassing the comprehensive examination of minute life forms, including bacteria, fungi, protists, and their behavior, evolution, ecology, biochemistry, physiology, and the pathology of diseases they have the potential to induce (Biologydictionary.net Editors, 2017). This broad scope highlights the significance of using diverse tools to interrogate the behavior and characteristics of specific microorganisms. The main focus of my research is the sliding motility of Clostridium perfringens. C. perfringens, previously known as C. welchii or Bacillus welchii, is a Gram-positive, anaerobic bacterium widely distributed in the environment (Ryan et al., 2004). This bacterium is a common inhabitant of decaying organic matter and gastrointestinal tracts of humans, vertebrates, insects, and soil. The bacterium is rod shaped, with parallel sides and rounded ends (Fig. 1). It measures 3-8 µm in length (Aryal, 2023). Figure 1: C. perfringens grown in Schaedler's broth, shown with gram-stain (From CDC, 1974). 1.2 Sliding Motility of C. perfringens Unlike most bacterial motility mechanisms that require designated cell appendages, such as a flagellum, sliding motility is a type of "passive" surface motility that is driven by increase of the colony mass due to growth and division of the bacteria (Mehdizadeh Gohari et al., 2021). The efficiency of bacterial sliding motility can be enhanced by a number of facilitating mechanisms, such as secretion of a mucous-like substance that reduces friction against the substrate surface and keeps the bacteria in 2D expansion (Holscher & Kovacs, 2017). In C. perfringens, the sliding motility is facilitated through maintaining cell-cell linkages and forming elongated chains (Liu et al., 2013; Liu

bacteria to expand in a 1D fashion and achieve notably high speeds. However, the inherent vulnerability of long chains to breakage, particularly under stress buildup, poses a critical challenge to the ultimate efficiency of the sliding motility (McMahon et al., 2022). To comprehensively understand the mechanisms facilitating chain growth, it becomes essential to explore additional factors that could contribute to this process. For instance, an intriguing observation is the tendency of each half of a broken chain to form prolonged lateral contacts with the other. This might provide mechanical support to the cells, making the chains more resilient and less prone to breakage. To investigate these facilitating mechanisms, it is essential to initially derive insights from the experimental data about how C. perfringens moves within

its colony. This process will provide the essential information that can either validate or refute certain assumptions regarding the mechanisms that support its motion. The imperative need for an automated image processing pipeline arises from the necessity to analyze a large volume of videos systematically, while ensuring accuracy and objectivity in the assessment of bacterial motility patterns. This approach not only expedites the analysis process but also contributes to the reliability and reproducibility of the study's outcomes. Integrating an AI-based approach is particularly advantageous as it enables the automatic detection of objects from complex backgrounds such as those found in microscopy images. The AI algorithms essentially identify and delineate the boundaries of bacterial cells. Appropriately tuned algorithms can achieve a precision at least comparable to human eyes, while avoiding human errors and biases and saving human labor. The same level of precision would be impractical to achieve using traditional computational algorithms. The results of the AI-based segmentation are then fed into the tracking algorithms to obtain continuous time trajectories of the cells' motility behavior. Finally, post-processing algorithms are used to extract from the time trajectories the key quantitative characteristics of the cells, such as velocity, growth rate, and division frequency. The overall image analysis pipeline streamlines the labor-intensive tasks associated with manual segmentation, tracking, and measurement, providing a more efficient and accurate analysis of C. perfringens' movements within colonies. 1.3 Machine Learning and Image Analysis

Artificial intelligence (AI) is **the** creation **of computer systems that can perform tasks that typically require human intelligence** like problem solving, **learning** 15

, and perception (Russell & Norvig, 2016) By processing large amounts of data and recognizing patterns within it, artificial intelligence can learn from experiences, adapt to new inputs, and carry out tasks with a degree of autonomy. It can automate tasks and process large amounts of data quickly and efficiently. Artificial intelligence is an umbrella term that includes several technologies such as

Machine Learning (ML), Deep Learning (DL), and Natural Language Processing (NLP) (Fig . 2). Figure **2** : Three-tiered hierarchy **of** 37

Artificial Intelligence (From Singh, 2018).

Machine Learning is a subfield of AI that focuses on developing algorithms and models that allow computers to learn and make predictions 20

based on the input **data**

. Machine learning systems use data to improve their performance over time. At their core, the

machine learning based **algorithms build models based on** the **sample data, known as** "training data," **to make predictions or decisions without** 31

human intervention. Training allows the model to make accurate

predictions or decisions when presented with new, unseen data 52

. Model training typically involves adjusting the model's parameters, particularly batch size, epochs, channels, to minimize the loss between the

predictions made by the model and **the actual** observed **values** of **the** 53

data. Batch size refers to the number of training samples used to train the model in one iteration. Additionally, the number of Epochs refers to

the number of **times the algorithm will work through the entire training dataset** 22

. Furthermore, channels refer to the depth dimension of input data that represents specific types of information within the dataset. For images, channels often represent color depth. Machine learning can be broadly classified into several types, each suited to different kinds of data challenges. They include

Supervised Learning, Unsupervised Learning and Reinforcement Learning (Fig. 3). Figure 3: Types **of Machine Learning** 42

Algorithms (From Piyush et al., 2020). Supervised learning involves the machine learning a particular function that connects inputs to their corresponding outputs, utilizing a dataset composed of input-output pairs as examples.

Supervised learning is typically employed when specific objectives need to be achieved based on given 35

inputs. Some of the approaches used in supervised learning include classification, regression, neural networks, naive Bayesian model and random forest model. Algorithms trained under supervised learning can predict future events with high accuracy by learning from past data. These algorithms can enhance their performance over time through learning from previous results (Rancho, 2021). Supervised learning is the most common type of algorithm used in today's world. The diagram in Figure 4 depicts the supervised learning process where a labeled data set consisting of various fruits serves as the raw input data. The data is provided to the machine learning algorithm, which then processes this information and learns to associate specific fruit types with their corresponding labels. The output consists of classified results where the model can now identify and categorize new data points effectively - for example recognizing an apple, a pear, or an orange. Figure 4: Diagrammatic depiction of how supervised learning works (From Rancho Labs, 2021). The most notable tasks in supervised learning include 'classification', which categorizes data into distinct classes, and 'regression', which aims to predict continuous outcomes based on input data. Examples of Classification-based learning include distinguishing between spam and non-spam emails, identifying a person from a digital image or video, commonly called facial recognition, and categorizing images into various classes such as identifying objects such as animals from the background in a photograph. On the other hand, some examples for regression-based learning are predicting weather conditions like temperature, rainfall and predicting future stock market prices. Figure 5: Comparison between Regression and Classification in Supervised Learning (From Raj R, 2024). Figure 5 distinguishes the two types of supervised learning-based approaches in the context of weather forecasting. On the left side of Figure 5, the question “what will be the temperature tomorrow?” refers to a regression problem where the goal is to predict a continuous numerical value showcasing the regression’s ability to provide a quantitative forecast. Regression models would take in various inputs like historical temperature data, weather conditions and produce a value that represents a prediction on a continuous scale. On the right side of Figure 5, the question “will it be hot or cold tomorrow” refers to a classification problem which is about predicting a category rather than a numerical value. In classification the model would assign the weather forecast to one of these discrete categories rather than predicting an exact temperature. Unsupervised learning involves instructing a machine using data that has not been labeled. The objective is to allow the machine to autonomously categorize the unordered data by identifying commonalities and patterns without the

benefit of previously labeled training data. Unsupervised learning algorithms are used in scenarios where previously labeled data is not available. They can be classified into clustering and association. Unsupervised learning has very few applications in the real world. They can be used in recommendation systems, fraud detection and so on. Figure 6: Diagrammatic depiction of how unsupervised learning works (From Rancho Labs, 2021). Figure 6 outlines an unsupervised learning process that begins with the input raw data which includes images of apples bananas and oranges. These fruits represent the data points that are fed into the algorithm without any associated output labels, indicating an unsupervised learning scenario. After the algorithm processes the data, the output is shown, where the model is now able to identify and differentiate between apples bananas and oranges. Reinforcement Learning is a method where a machine is trained to make a sequence of decisions to maximize the probability of getting a reward. The algorithm's decisions are influenced by the receipt of rewards for positive outcomes and penalties for negative ones. Unlike supervised learning which relies on training models with pre labeled data, reinforcement learning focuses on acquiring knowledge through the outcomes of previous actions and feedback. Some of the

reinforcement learning algorithms include Q learning, temporal difference, Monte Carlo tree search and **asynchronous actor critic agents** 32

(Piyush et al., 2020). The machine continuously performs actions, receives rewards (feedback) from the environment and uses this information to refine its approach (Fig. 7). This cycle repeats until the machine’s performance meets the predetermined level of satisfaction. Figure 7: Diagram illustrating the concept of reinforcement learning (From Piyush et al., 2020). Figure 8: Flowchart of the types and sub-types of Machine Learning Algorithms (From Rajbanshi, 2021). Figure 8 summarizes

the three primary categories of machine learning: supervised learning, unsupervised learning, and reinforcement learning 39

. It also depicts the subgroups of each category. In a nutshell, machine learning algorithms function by learning from data. Data is gathered and preprocessed to ensure that it is in a usable format. An algorithm is selected based on the task to be performed. The algorithm is trained on a subset of data called the ‘training data set’ where the algorithm learns from its features and corresponding outcomes. The model is then evaluated using a separate data set called the ‘testing data set’ to test its predictive performance. Machine learning has found increasingly common use in image processing where computers are used to identify objects within images and videos. Image processing is broadly defined as any method that is used to perform certain operations on images in order to extract useful information from them or improve their quality (Yadav, 2024). Basic image processing tasks, such as enhancing image contrast, can be carried out using classical algorithms. However, more complex tasks like image classification, object detection, and segmentation often require the sophisticated capabilities of supervised learning algorithms. For tasks like image classification, object detection, and segmentation, supervised learning algorithms are used. 1.4 Machine Learning-based Image

Segmentation Image segmentation refers to dividing an image into multiple regions (segments) called objects to simplify the representation of an image. The objects encompass a set of pixels that are assigned the same label based on specific characteristics or features (Tyagi, 2013). For instance, for an image containing a landscape (Fig. 9), pixels with similar values were labelled 'sky' for the blue portion of the image, 'green' for trees and so on. Through labeling and categorizing the pixels in an

image, image segmentation is used to locate **objects and boundaries** 51

within images. Figure 9: Example of Image Segmentation of a landscape image (From Klingler, 2024). There have been many applications of image analysis in the field of biological sciences. One of the developments I would like to highlight is the implementation of Deep Neural Networks for early detection of lung cancer through the image analysis of MRI scans (Jakimovski & Davcev, 2018). The AI-based image segmentation algorithm was trained through extensive datasets, involving MRI scans categorized by medical professionals into cancerous and noncancerous groups. The trained DNN demonstrated an increasing accuracy in identifying lung cancer as the tumor size increases from stage 0 to stage 4. For example, Figure 10 includes CT scans for lung cancers from stage 0 to stage 4, showing the progression from a small, localized tumor to cancer that has spread throughout the body. The certainty of detection by DNN increases to 81% for stage 4 cancers. This indicates that the network can effectively recognize cancer related patterns in MRI scans as the disease progresses. The use of DNNs in medical imaging showcases what is achievable with advanced image segmentation. Figure 10:

Results of classifying slow progressive lung cancer images using DNN 45

. The percentages represent the model's confidence levels in diagnosing the stage of lung cancer, with the value increasing from Stage 0 to Stage 4 as the tumor size and spread become more pronounced (From Jakimovski et al., 2018). 1.5 Machine Learning-Based Image Processing for My Project The overarching goal of my research is to develop a comprehensive understanding of C. perfringens sliding motility through detailed image analysis. In pursuit of this aim, image segmentation will play a pivotal role. Leveraging AI-driven algorithms, the project will segment high-resolution microscopic images to isolate and identify individual bacterial cells. This segmentation serves as the foundation for tracking cell movement, enabling the precise characterization of motility patterns. The results will be analyzed and key motility characteristics like cell velocity, cell growth rate and division frequency will be quantified. These analyses will provide insights into the dynamic nature of the cell chain-mediated sliding motility in C. perfringens. The data will also provide information to validate and help refine the mechanical model developed in my advisor's lab (McMahon et al., 2022). Previous AI models for image segmentation have laid the groundwork for sophisticated analysis in various fields. This research relies on

the ability of these models to adeptly manage and analyze complex image datasets. By adapting and improving upon these existing models, the research aims to achieve a level of precision in segmentation. The results of the segmentation will be used to track the movement of cells and extract information of interest. In the following paragraphs, I will review several published models for cell image segmentation and discuss their relevance to my research project. In 2022, Padovani et al. introduced

an open source, (Graphical User Interface) GUI-based software **framework** called **Cell** 46

-ACDC (Padovani et al., 2022). This tool is designed to enhance the analysis of live-cell imaging data to the application of advanced deep learning models for cell segmentation, tracking, and cell annotations. The software encompasses deep learning models like StarDist, Cellpose, YeaZ for precise single-cell segmentation. A key feature of this tool is its user-friendly GUI that allows users to correct segmentation, track errors and annotate cell cycle stages interactively. The framework is mainly applied for understanding different stages of the cell cycle. However, currently the deep learning models within Cell-ACDC are optimized for specific types of cells such as yeast and mammalian cells and do not work very well for rod- shaped bacterial cells. Another image analysis tool used for image analysis of cells

is CellProfiler (Carpenter et al., 2006). It **is an open-source software** tool **designed for high-throughput image analysis of** 5

cells, which enables analysis of large quantities of complex image data. CellProfiler provides quantitative analysis of various complex phenotypic features of the cells, such as protein localization patterns and organelle morphology. It also provides a user-friendly, modular framework that allows researchers to customize their analyses. However, challenges remain in accurately identifying cells in crowded images, which necessitated exploring additional tools to meet the specific needs of my project. In pursuit of a more tailored solution, I experimented with LivecellX, a promising AI-based image segmentation and tracking pipeline (Wang et al., 2019). This pipeline exploits the faster Region with Convolutional Neural Networks (R-CNN) and the watershed algorithm. The two methods are combined to get optimal results. The resulting pipeline works well on various types of input images, including fluorescent, phase contrast and differential interference contrast images. When compared to traditional pixel-wise classification, their pipeline achieved higher cell count accuracy. Unfortunately, LivecellX proved suboptimal for my project needs. It failed to detect several C. perfringens cells in almost every image frame, making subsequent cell tracking and data analysis impossible. This failure is attributed to the algorithm being trained on datasets that are not representative of rod-shaped bacteria, leading to considerable under- segmentation in my data set. This is further explained in chapter 2 in detail. Finally, SuperSegger, a MATLAB-based image processing and analysis package, demonstrated better performance in handling images

of rod-shaped bacteria (Stylianidou et al., 2016). SuperSegger is equipped with machine-learning algorithms to refine cellular boundaries and is capable of segmenting colonies with high cell densities. This tool

is particularly well suited for high throughput , quantitative **analysis of** live- **cell** fluorescence images **of**	25

bacterial cells. SuperSegger's robustness in accurately detecting and segmenting C. perfringens cells aligned well with the objectives of my research, effectively overcoming the limitations experienced with the previous framework. Additionally, the software includes automated tools to resolve common errors in cell tracking, which is essential for studying cell lineage and dynamics over time. It also

provides a range **of post-processing tools for single-cell analysis**	49

. SuperSegger was able to accurately identify cells and monitor them across a series of frames. The comprehensive post-processing tools allowed me to observe and quantify cellular behaviors with ease. Overall, the implementation of SuperSegger within my project framework not only addressed the initial challenges faced with LivecellX, but also provided a level of detail and accuracy that was important for the success of my research objectives. To illustrate the complete image analysis process utilized in my research, the overall procedure is diagrammatically represented in Figure 11. It lays out each step of the workflow, from initial image collection through processing with SuperSegger to the final analysis, providing a visual summary of how all the techniques were integrated to achieve the project's goals. Figure 11: Flowchart of the undertaken tasks outlining the sequence of steps in my research. In Chapter 2, I will demonstrate the methodologies I started with. These were the methods that I employed that did not produce the anticipated results. Chapter 3 will then focus on the approaches that proved to be effective. CHAPTER 2: INITIAL RESEARCH STRATEGIES 2. 1 Data Acquisition A critical early focus of my research involved creating a well-annotated dataset from the microscopic videos of C. perfringens. Recognizing that the quality of the data would significantly impact the final analysis, I prioritized obtaining clear and distinct images. These would serve as a robust foundation for the model to perform accurate and precise analysis. Given that I did not have access to the original tiff files in the beginning, I needed to run some preprocessing on the video. Furthermore, I employed the LabelImg tool for data annotation which facilitated the annotation process by allowing for the labeling of individual C. perfringens cells within experimental videos. 2.1.1 Methods The videos (Fig. 12) capture and provide a continuous record of the dynamic movement of cells, enabling precise analysis of the motility patterns of the bacteria. These videos were taken by Dr. Stephen Melville's laboratory. The dataset is comprised of a video file, with a playback duration of approximately 22 seconds. The video, initially recorded over a span of 60 minutes, required conversion into a format suitable for detailed frame-by-frame analysis. To achieve this, the video was first transformed into 106 individual image frames, yielding a frame rate of approximately 0.0294. The

original .mov files were converted to .avi format using MPEG_Streamclip with the QuickTimeinstaller plugin. This conversion was necessary because the .avi format is more compatible with various image processing tools and allows for more straightforward manipulation and analysis of video data. However, it is important to acknowledge that this format conversion process can lead to some loss of data. Converting from one video format to another can potentially reduce the quality of the images due to compression and other factors, which might affect the precision of the subsequent image analysis. Following this, the .avi files were processed using ImageJ software to extract individual image frames, as images are needed as inputs for the segmentation algorithm. This stepwise conversion ensured that the video data was in the optimal format for subsequent image processing tasks. Figure 12: Video of C. perfringens cells obtained from Dr. Melville's Lab. In the subsequent section, detailed information is presented on the production of the final image frames. Initially, the annotation process focused on a subset of 10 images, followed by a preliminary test on 3 selected images. Upon achieving satisfactory results in the segmentation and tracking of these 3 images, the dataset was planned to be expanded. Specifically, 76 images would be incorporated into the training set, while 30 images would be added to the testing dataset. This stepwise approach would ensure a gradual refinement of the AI-based segmentation model, optimizing its performance before scaling up the dataset for more accurate results. The annotation process provided the training dataset for the machine learning model. This dataset allows the algorithm to learn the visual cues essential for identifying the shape and boundaries of individual cells. A combination of tools and methods was employed to ensure precise and effective annotations. Initially, while tools like Fiji are generally used to enhance image contrast before annotation, the C. perfringens videos already exhibited sufficient contrast, eliminating the need for this preliminary step. I chose to use the LabelImg tool for its robust features, which include various shapes of bounding boxes. This allowed me to assess how different shapes influenced the annotation quality and to test the tool's accuracy in defining cell boundaries. The ease of drawing these bounding boxes with LabelImg significantly facilitated the annotation process. The annotation process hence involved the following steps: • I manually drew bounding boxes around the cells of interest using LabelImg. • I utilized the polygon tool within LabelMe to outline the boundaries of the cells and labelled each cell as 'cell.' Various tools like the rectangle, line, point was tried for annotation, but the polygon tool is chosen for its flexibility in accurately capturing the irregular shapes of the cells. Moreover, subtle details in the annotation process, such as drawing profiles closer to the outside of the cell boundary were considered. Drawing profiles closer to the inside of the cell boundary yields more precise segmentation outcomes. Despite the precision that tools like the polygon tool offer in capturing the irregular shapes of cells, the subjective nature of manual annotations can lead to slight variations in how boundaries are defined. Moreover, drawing annotation profiles slightly outside the cell boundary, although generally resulting in more accurate segmentation, still depends on the annotator's judgment and consistency. These factors can lead to minor discrepancies in the data, potentially affecting the robustness and reliability of the training data set. Additionally, any inaccuracies in manual annotation might not only lead to errors in current analyses but could also propagate through subsequent stages of data processing, impacting overall study outcomes. After completing the annotations, the data was saved in JSON format. All individual JSON files were then merged into a single comprehensive COCO format JSON file, consolidating all data into one structured format which is highly advantageous for subsequent image segmentation. 2.1.2 Results The aim was to create a high-quality annotated dataset, with each image containing precisely drawn bounding boxes for each cell. The dataset was intended to be divided into two parts: 10 images

for the training dataset and 3 images **for the testing dataset. The** 22

goal was to start small and then expand as we achieved satisfactory results. However, during the data acquisition process, we anticipated potential challenges that could affect the quality of our results and several alternative strategies were considered with respect to changes in the quality of the input image, annotation consistency and sample size. 2.2 Segmentation of Individual C. perfringens Cells The primary goal of my research was to develop a reliable method for identifying individual C. perfringens cells in microscopy images. This required the use of an AI-based segmentation tool to isolate and segment cells within each image frame effectively. Initially, I utilized a segmentation pipeline developed by (Wang et al., 2019) which was designed to accurately determine cell boundaries and efficiently segment them. Accurately segmenting individual C. perfringens cells is crucial for this research. The ability to precisely outline and separate each bacterial cell in the experimental videos is essential. This high level of accuracy in segmentation lays the groundwork for in-depth quantitative analysis, allowing for the measurement of key cellular parameters such as velocity, elongation rate, and division frequency. Each of these metrics is vital for understanding the complex biological behaviors of the bacteria under study. 2.2.1 Methods To prepare the images for analysis, a series of preprocessing steps were implemented. Firstly, the images were normalized to grayscale using OpenCV to streamline the subsequent analytical processes. Standard preprocessing techniques were then applied to enhance the overall quality of the images. The segmentation employed in this project used the Cellpose model, which is a deep learning segmentation model that can

segment cells from a wide range of image types 54

. To enhance its performance and precision, the model was fine-tuned to optimize crucial parameters, including batch size, channels, and epochs. In the assessment of the Cellpose model's segmentation efficacy, a random selection of images from the dataset was used to display the results. Both the original raw images and the corresponding segmented images were compared to analyze the model's performance, allowing for a qualitative evaluation of its segmentation accuracy and highlighting areas for potential refinement or improvement. Code Details: The script implements the Cellpose machine learning model and begins by importing the necessary Python libraries such as Cellpose, NumPy, OpenCV, and Matplotlib. Cellpose is an advanced machine learning tool designed for the segmentation of cells from microscopic images that is specifically built on deep learning mechanism. It initializes a 'cyto2' Cellpose model configured to leverage GPU acceleration for efficient processing. The model training phase involves loading and preprocessing of image data from specified directories, where the raw cell images are converted to grayscale and normalized—a prerequisite step for deep learning models. Importantly, the segmentation labels are generated using the ‘skimage.measure.label’ function, which assigns unique identifiers to each cell. For training, the script organizes the image data and the corresponding masks by reading them from specified

directories and ensuring they are properly synchronized—each image has a matching mask. The images are preprocessed by converting them to grayscale and normalizing them, which standardizes the input data and helps the model learn more effectively. Cellpose's training functions include parameters such as batch size, the number of epochs (iterations over the entire dataset), and channels. These parameters control how the model learns, impacting both the speed of training and the quality of the model's predictions. A snapshot of the code segment where the training of the machine learning model occurs is provided below. 2.2.2 Results In the anticipated outcomes (Fig. 13) of this segmentation experiment, the results show three crucial components: the original raw images, the Cellpose-generated masks representing the model's predictions, and the label masks derived from manual annotations. The comparison of these elements reveals the efficacy of the Cellpose model in accurately segmenting individual cells compared to the annotations. Figure 13: Example results of segmentation. Left: Raw image. Middle: Segmented cells after training. Right: Label masks obtained after annotation. In this example, the cells in the lower right corner of the image were missed by the Cellpose model, indicating further need to train the model or adjust its parameters. While the segmentation model is expected to provide comprehensive results, it may not accurately capture every cell (Fig. 13, middle). There were instances where the model falls short in detecting specific cells within the images. Recognizing these missed cells was crucial for refining and improving the segmentation model. The analysis of such discrepancies means there is a need for further enhancements like increasing the number of iterations in training and expanding the size of the dataset or using alternate software and tools to carry out image segmentation. Furthermore, my plan was to evaluate Wang's cell tracking algorithm (Wang et al., 2019) within the pipeline, where each cell is assigned a unique identifier. It begins by extracting video frames and masks from the samples, which are then combined to create annotated frames for tracking. The tracking is conducted using the SORT (Simple Online and Realtime Tracking) algorithm, a popular choice for its efficiency in handling real-time tracking scenarios. This algorithm processes the combined frames and masks to associate each detection (cell seen in the frame) with existing tracked objects. Additionally, the visualizer in the pipeline was expected to display the assigned identifier on each cell, providing a clear representation of their trajectories across frames. However, given the model's tendency to miss numerous cells, accurate tracking was unattainable. In the next section I will discuss the alternative strategy I employed that effectively addressed these challenges and led to successful outcomes. CHAPTER 3: FINAL RESEARCH STRATEGIES 3. 1 Task 1: Data Acquisition Tiff images from Dr. Melville's laboratory were analyzed (Fig. 14). TIFF (Tagged Image File Format) files, denoted with a .tif or .tiff extension is a type of graphics image file format that are used for their ability to store high quality images with rich details. Additionally, TIFF files contain metadata information such as image dimensions, resolution, and timestamps. Figure 14: TIF files obtained from Dr. Melville's lab showing the original file names, dimensions and size. I obtained 121 TIF images that contained the bacterial size of interest. Since the total recording time was 60 minutes (3600 seconds), the frame rate is approximately 0.033 frames per second or 2.01 frames per minute. This means that roughly 1 frame was captured every 30 seconds during the 60-minute recording. The images should be viewed using a software like Fiji. Fiji is a variant of the widely used open-source program ImageJ, particularly used for the analysis of biological images (Schindelin et al., 2012). Alongside these images, I also gathered metadata for 24 distinct cells which included comprehensive details such as time, intensity, photosensor reading, exposure time, excitation filter, emission filter, neutral density filter settings and stage coordinates. A sample image is shown in Figure 15. Figure 15: A sample frame from the microscopy video 3. 2 Task 2: Segmentation of Cells I employed a technique for distinguishing individual C. perfringens cells in the tiff images. This required separating and segmenting the cells in each image frame by employing an artificial

intelligence-powered segmentation tool called SuperSegger. SuperSegger is an advanced MATLAB-based package developed by the Wiggins Lab at the University of Washington for automated cell segmentation, fluorescence quantification, and analysis, tailored specifically

for high-throughput time-lapse fluorescence microscopy studies **of** in vivo bacterial **cells** 25

. It is particularly noted for its efficiency in handling rod-like bacterial cells, making it an ideal tool for researchers focusing on cell-cycle dynamics, protein complex analysis, and cellular morphology studies. Utilizing machine learning algorithms to optimize cell boundary identification for various purposes, the software is highly effective in segmenting cells that are in contact. Furthermore, SuperSegger can track cells across successive frames and applies error correction to ensure accurate segmentation and lineage tracking over time. Moreover, SuperSegger offers a wide array of post-processing tools to allow in-depth analysis of cellular behavior and properties. The latest iteration of SuperSegger integrates the Omnipose segmentation algorithm, enhancing its segmentation accuracy. This updated version, SuperSegger-Omnipose, has an even more refined analysis, especially for cells in close contact or undergoing complex growth patterns. Omnipose is a tool for image segmentation that enhances the capabilities of Cellpose. Cellpose

is a software **tool that** utilizes **deep learning techniques to** segment cells from **images** 16

(Stringer et al., 2021). Developed as an extension and improvement over existing models like Cellpose, Omnipose addresses several limitations observed in previous methods when dealing with complex cell shapes and closely interacting cells. The strength of Omnipose lies in its unique network outputs and an innovative combination of

a cell boundary probability map , a **distance field, and a flow field** derived from **the gradient of the distance field** 18

. These elements allow Omnipose to precisely map out cell boundaries even when cells are densely packed or exhibit irregular shapes. One of the major innovations in Omnipose is its use of the gradient of the distance field, which enhances its ability to handle cells of diverse and complex morphologies, ranging from simple spherical bacteria to more elaborate branched forms (Cutler et al., 2022). SuperSegger offers a user-friendly interface with a GUI for different functionalities such as image segmentation, result viewing, and parameter training. It also provides non-GUI alternatives for command-line operation. Key functionalities include processing

and alignment of fluorescence and phase images, segmentation of cells, tracking and linking cells across frames, and generating detailed output files that include cellular descriptors and lifetime data. Additionally, SuperSegger aligns images to account for stage drift, ensuring that cell tracking remains consistent throughout the experiment (Stylianidou et al., 2016). Figure 16: Comparative Visualization of Bacterial Cell Segmentation: Unprocessed Phase-Contrast Image and Automated Segmentation Outcomes by SuperSegger and SuperSegger 2, Showcasing Enhanced Algorithm Precision (From Wiggins Lab, 2024). In Figure 16, the image illustrates the capabilities of the SuperSegger software in segmenting bacterial cells. On the left is the original phase-contrast microscopy image of a cluster of bacteria. The middle and right images show the segmentation results from two different versions of SuperSegger. The middle image, labeled ‘SuperSegger’, has the cell boundaries highlighted in green, showing the segmentation outcome of the initial version of the software. On the right, the ‘SuperSegger 2’ version is the latest model that incorporates Omnipose in it. Both versions are apparently capable of recognizing and delineating individual bacterial cells, even when they are tightly packed together. The distinction between the two figures primarily lies in the finer details of cell boundary delineation. With the original SuperSegger, close inspection reveals some irregularities and distortions along the boundaries—it's evident that the outlines are not uniformly smooth. The distortions in the cell boundaries can lead to a loss of data accuracy, which is particularly detrimental for downstream analyses requiring precise cell measurements. In contrast, SuperSegger 2, which integrates the Omnipose algorithm, achieves much smoother and more consistent outlines, with less distortion at the cell boundaries. This precision ensures a more accurate representation of each cell's shape, which is crucial for detailed analyses that require high precision in cell segmentation. 3.2.1 Methods The process was executed through a series of systematically planned steps, which I will detail in this section. Step 1: Initial setup and name conversion: before beginning to work with SuperSegger the image files were properly named and formatted. According to SuperSegger, the images should be monochromatic, in .tif format and be captured at either 60x or 100x magnification. This detail is crucial because SuperSegger’s segmentation parameters are tuned to these specifications. Since the images were not named according to SuperSegger’s required naming convention, I used the GUI to convert the file names accordingly this convention follows the format [basename]t[number]xy[number]c[number].tif, where 't' indicates the frame number, 'xy' the timelapse position number, and 'c' the channel number. Figure 17 shows the GUI for SuperSegger. There are fields for inputting parameters to convert names to the required naming convention for the segmentation software. Here's a breakdown of what each term means: Basename: This is the name that we want to prepend to all our images, which usually includes the date of the experiment and the bacterial strain. In the sample video it is ‘str-13’. ChannelNames: array of strings that indicate the different channels in the filename, which is ‘w- 50’. Time Prefix: This string precedes the time frame number in the current filenames. For instance, if the filename is 'exp1_time5.tif', the time prefix would be 'exp1_time'. For the sample video, since the original file name contains ‘_t001’, the prefix will be ‘_t’. Time Suffix: This is the string that follows the time frame number in the current filenames, which is the file extension ‘.tif’ in the sample video. xy prefix: These are the characters in the current filename before the number that indicates the xy positions. xy suffix: These are the characters in the current filename after the number that indicates the xy position. The xy prefix and suffix are not mandatory parameters and can be skipped. Furthermore, Figure 18 shows the names of files after the conversion is done. SuperSegger retains a copy of all the initial images, preserving their initial filenames, in a directory named ‘original’. Figure 17: SuperSegger GUI for converting image filenames into appropriate formats for SuperSegger to be able to read it. Figure 18: The updated names of the files after applying the name conversion through the SuperSegger GUI. . Step 2: Running the Segmentation I

began by directing the program to the data directory and setting the segmentation parameters to the default values. SuperSegger then processed each image, aligning them over time, distinguishing cell regions against the background, and connecting these regions across subsequent frames to monitor individual cells. SuperSegger automates cell segmentation by first separating cell clusters from the background using a consistent thresholding technique. The watershed algorithm then detects individual cells within these clusters. These segments are refined by neural networks that classify cell boundaries and optimize the segmentation, ensuring that each cell is accurately identified and represented in the final analysis. To address the issue of stage/frame drift during extended time-lapse imaging, which can cause a significant shift in the imaging field of the video, SuperSegger employs an efficient alignment algorithm. This algorithm uses cross-correlation to achieve precise frame alignment down to subpixel accuracy (Stylianidou et al., 2016). Initially, I performed segmentation solely with SuperSegger, followed by a subsequent segmentation using SuperSegger enhanced with Omnipose. An additional step required was the installation of Omnipose and the activation of its environment via the command line before the segmentation process. During the segmentation process, I also had to enter the Omnipose command, provided by MATLAB, into the terminal, and then wait for Omnipose to complete the segmentation of images and the creation of masks. 3.2.2 Results Figure 19 showcases the segmentation results using only SuperSegger. It is evident that numerous cells exhibit errors, and the boundaries appear somewhat undefined. Figure 19: Segmentation result of SuperSegger. Figure 20 illustrates the segmentation outcomes when SuperSegger is used with Omnipose. There is a noticeable improvement compared to using SuperSegger alone, with minimal errors observed in the results and the cell boundaries being much more clearly defined. Although some of the cells are marked as errors, SuperSegger still considers them to be ‘cells’ and assigns correct IDs to them. This segmentation performs better than the previously employed methods. During the segmentation process, some impurities in the image were mistakenly identified as cells by the segmentation algorithm. These impurities could be debris or non-cellular material within the imaging field. I excluded these objects based on their area, removing those that fell outside the typical size range of the cells being studied. This approach was effective in mitigating the inclusion of erroneous data, allowing for a more accurate analysis of cell motility. By setting a threshold for the area, I was able to filter out these non-cell entities, ensuring that the subsequent analyses were performed only on true bacterial cells, thus enhancing the reliability of the results. Figure 20: Segmentation result of SuperSegger-Omnipose viewed from the SuperSegger GUI. Figure 21: Comparative analysis of cell processing showing the original image (left), boundaries of the cells predicted by SuperSegger (middle) and the corresponding masks (right). Figure 21 demonstrates the steps of the processing system. The first image shows the unprocessed, original phase contrast microscopy image. The second image depicts the predicted outlines of each bacterial cell. The red lines indicate the boundaries of each cell as determined by the algorithm. The third image shows the predicted masks which have distinct labels (marked by different colors). 3.3 Task 3: Cell Tracking The main aim of employing cell tracking is that it allows us to understand various processes, such as cell division, migration, and interactions. Cell tracking facilitates quantitative analysis of cell behavior, providing data on parameters that we can use later to calculate sliding characteristics. The core of the tracking is the linking algorithm, which uses a cost function to evaluate potential links between cell regions across consecutive frames, based on spatial overlap, centroid distance, and area changes. This algorithm selects the linkage with the minimum cost, ensuring accurate tracking of each cell’s trajectory. By continuously applying this method throughout all frames, SuperSegger tracks cell lineage and history, recording detailed data on each cell’s properties at every time point (Stylianidou et al., 2016). The software links cell regions from frame to frame using their IDs. Each cell detected by the segmentation

process is given a unique ID. This ID is consistent across the frames, meaning that a cell will have the same ID throughout the frames as it moves or divides. This allows for the precise tracking of cell lineages and behaviors over time. In the process of using SuperSegger for cell segmentation in a series of 121 images, a challenge emerged due to the software's mechanism of assigning unique IDs for cell tracking. This tracking mechanism was not equipped to handle the assignment of IDs to a large number of frames simultaneously. However, the tracking performed well when segmented in batches of 10 frames at a time. Therefore, I segmented the images into smaller subsets, specifically in batches of 10 images and linked these segmented batches with consistent ID assignments. 3.3.1 Methods To ensure continuity and accuracy in cell identification across sequential microscopy images, I employed a backward mapping strategy focusing on meticulously designated overlapping frames. These overlapping frames, established every ten frames (e.g., 11L and 11N, 21L and 21N), are specifically arranged such that 'L' (Last) represents the last frame of one segment and 'N' (Next) signifies the first frame of the following segment. The primary goal of these overlaps is to align identical bounding box coordinates across the overlapping frames, facilitating the precise matching of the same cells and thereby providing a reference point for confirming cell identities. In practice, the identification process between overlapping frames involves a direct comparison of bounding boxes. For each cell in frame 111N, its bounding box is compared to those in 111L. If the bounding boxes are exactly the same, it confirms that they represent the same cell. This ID update is then propagated backward through the previous frames. For instance, if a cell identified as ID 2 in 111N is matched with what was originally labeled as ID 1 in 111L, the algorithm systematically updates the ID from frame 111L backward to frame 101N, replacing all instances of ID 1 with ID 2. This action guarantees that the cell's identity remains consistent across frames. The process repeats for each subsequent set of overlapping frames, such as 101N and 101L. Using the updated IDs from the most recent segment as the reference, I continued matching cell IDs in overlapping frames and update IDs backward through the intervening frames up to the next overlap. This method of backward propagation of cell IDs ensures systematic correction of cell identifications across the entire image sequence. SuperSegger provides a GUI for users to visualize the unique IDs assigned to each cell and visually follow these cells across frames. The ID field should be checked as shown in Figure 22, to show the unique identifiers assigned to each cell. Figure 22: The GUI of SuperSegger showing the tracking field on the left and the corresponding frame on the right. 3.3.2 Results The image (Fig. 23) showcases segmented bacterial cells, each assigned with a unique identifier (ID), allowing for individual tracking and analysis across various frames and time points. Figure 23: Segmented cells and their unique IDs as viewed from the SuperSegger GUI. 3.4 Task 5: Extraction of Key Characteristics of C. perfringens Sliding The aim of this task is to track and analyze the sliding motility of the bacteria over the duration of the microscopy video. Utilizing the output data downloaded from SuperSegger, I developed computer code to estimate the cell velocity, growth rate and division frequency of C. perfringens. 3.4.1 Methods Growth Rate: Two distinct growth rates were calculated to capture the full scope of C. perfringens cellular dynamics: the Instantaneous Growth Rate (IGR) and the Mean Growth Rate (MGR). Both characterize the rate at which bacterial cells change their size and grow in length. The IGR provides a detailed view of cellular growth at specific time points. It was calculated based on the change in length of the cell between consecutive frames (Eq. 1). …… Eq:1 Where: CellLength (ti) is the length of the cell at time point ti. CellLength (ti+1) is the length of the cell at the subsequent time point ti+1. ti+1 and ti are times of two consecutive frames. IGR was calculated for each consecutive time point across the observation period, yielding a time trajectory of growth rates that reflect how the cell's growth dynamics changes over time. Transitioning from these discrete instances to a broader perspective, the MGR assesses the average growth over the entire video. The MGR is an aggregate measure,

encompassing the overall growth trend from the start t0 to the final time point tfinal (Eq. 2). ……. Eq:2 Where: CellLength (t0) is the length of the cell at the initial time point t0. CellLength (tfinal) is

the length of the cell at the end of

time, tfinal. t0 is either the start time point of the video, or the start time point of a cell if it is born or moves into the imaging field in the middle of the video. tfinal is the end time point of the video, or the end time point of a cell if it dies or divides or moves out of the imaging field in the middle of the video. The MGR represents the cell's general growth performance over time. The average can be further taken across different cells to yield an estimate of growth rate for the colony or the strain. Velocity: The cell velocity will provide the speed at which individual bacterial cells move across a surface. It quantifies the rate of bacterial displacement and is often measured in micrometers per hour or a similar unit. The instantaneous cell velocity at each time point is estimated through calculating the displacement of the center of mass of individual cells over consecutive frames (Eq. 3). …… Eq: 3 Where: x(ti) and x(ti+1) are the x-coordinates of the position at times ti and ti+1, respectively. y(ti) and y(ti+1) are the y-coordinates of the position at times ti and ti+1, respectively. ti+1 and ti are consecutive time points between the two measurements. In the following paragraphs, due to the complexity of my system which involves numerous variables such as `rcm` and `r_center`, I have employed simplified notations to enhance clarity and focus on the key aspects of the analysis. This approach ensures that the descriptions and calculations remain accessible and understandable. I used two specific data points extracted from segmented bacterial cell images: r_center (Fig. 24), which represents the geometrical center of a cell, and rcm, which denotes the center of mass of the cell. The geometrical center, or r_center, is the midpoint of the cell's shape, calculated as the average position of all the points that make up the cell's boundary. It does not account for how the mass is distributed within that shape. In contrast, the center of mass, or rcm, was calculated based on the distribution of mass across the cell. In a uniform system where mass is distributed evenly, both measures will coincide. Figure 24: Image depicting the position or r_center of the cell (From Wiggins Lab, 2017). It is observed that r_center and rcm typically yield very close values for the data, which implies that the cells can be treated as having uniform mass distribution for the purposes of my analysis. Subsequently, I calculated the average of r_center and rcm to represent the cell's central position in order to maximize the accuracy. From this averaged central position, I calculated the displacement of the cell over time, which describes the change in position of the cell between two time points. This was followed by calculating the velocity by dividing the displacement by the time interval between all the time points. The step-by-step process is described below. 1. Calculating Average Position of the Cell for Each Frame: For each cell in each of the frames, I calculated the average of the values of `r_center` and `rcm` at time i (Eq. 4). …… Eq: 4 Where: xrcm(ti) and yrcm(ti) are the x and y coordinates of the center of mass at time ti, respectively. xr_center(ti) and yr_center(ti) are the x and y coordinates of the geometric center at time ti, respectively. 2. Determining Cell Displacements and Velocities Over Each Frame: To determine the displacement of a cell between two consecutive time points, the Euclidean distance formula was applied (Eq. 5). ……. Eq: 5 Where: xavg(ti) and xavg(ti+1) are the averaged x-coordinates of the cell's position at consecutive times ti and ti+1. yavg(ti) and yavg(ti+1) are the averaged y-

coordinates of the cell's position at consecutive times ti and ti+1. Velocity is the displacement per unit time. The time interval between frames (Δt), is used to calculate the velocity by dividing the displacement by `Δt`. ……Eq: 6 Where: v(ti) represents the velocity of the cell between the time points ti and ti+1. D(ti) represents the displacement of the cell between the time points as calculated in the previous step. Δt is the time interval between the time points ti and ti+1. Division Frequency: By definition, the cell division frequency can be determined through counting the number of cell divisions throughout the frames and normalizing it against the cell number and lapsed time. However, there are a few complexities in this naïve approach. In any living system, the number of cells can change dramatically over time, which is also true in my dataset. This variability complicates the direct calculation of division frequency because the number of cells is not constant. Determining which cell number to use --- whether it be initial, final, or average over time, impacts the division frequency calculation significantly. Each choice could yield different results and introduce biases. Given the above complexities, a more refined statistical approach is required rather than a straightforward ratio of divisions to cell numbers. In the statistical approach, a probability distribution for the waiting time for cell division, i.e., the generation time, must be assumed, in order for the estimation algorithm to be determined. In this work, I assumed an exponential distribution. Although the generation time of a bacterium generally does not follow the exponential distribution (Powell, 1956), the most common continuous population growth models essentially assume an exponentially distributed generation time. Therefore, the division frequency that is estimated based on the assumption of exponentially distributed generation time provides exactly the parameter wanted in the population growth model. The method for estimating the division frequency from the data is based on the memory-less property of exponentially distributed waiting time for a stochastic event. This memory-less property entails that for any sufficiently small time interval, Δt, the probability for the event to occur is proportional to Δt, regardless of how long the wait has lasted. Based on this nice property, one can estimate the probability for a cell to divide over the time interval between two consecutive frames through counting the number of state transitions and calculating their ratios. Specifically, each cell within the observed population can exist in one of the three distinct states in any given frame. 1. Living state (state 0): Cells in this state are alive under the observed conditions; 2. Dead State (state 1): This state represents cells that have died due to various factors; 3. Division State (state 2): Cells in this state undergo division where it divides into two daughter cells. The division frequency will be calculated by using the number of transitions that lead to division (state 0 to state 2) and the number of transitions from state 0 across all the observations in the data set. …….. Eq: 8 An example of the transition counting process is below. Example Cell 1: Consider the state sequence across frames as '0 0 0 0 2'. Here, the cell remains in the living state (0) for four frames and then transitions to the dead state (1) in the fifth frame. To count the number of transitions for cell 1, the changes between consecutive frames are observed. For Cell 1, the number of transitions from state 0 would be 4 and the number of division transitions is 1. Because cell division occurs at highly random times and most cells do not even experience a division within the timeframe of the video, to obtain more accurate estimation of the division frequency, I pooled the data from all cells. For each cell, the count of every progression from one frame to the next (as a transition) is recorded. This method captures the full transition of the cell's behavior over time, providing comprehensive data for statistical analysis. To calculate the number of transitions for each cell in the dataset, I used a straightforward method by subtracting the birth frame number from the death frame number (last frame is referred to as death frame in SuperSegger) for each cell. This calculation effectively counts how many frames each cell was observed providing a measure of how many transitions occurred during its observable lifecycle. In the example above, Cell 1 has a recorded birth frame number of 1 and death frame

number of 5, the number of transitions will be: death frame number – birth frame number = 4. By calculating the transitions this way, the total number of transitions in the dataset is retrieved. In order to calculate the number of transitions that lead to division, I first identified the division events for each cell. This can determine whether a division has occurred by checking if the divide flag is set to 1 in the cell's data. divide: 1 if successful division was observed. divide: 0 if no division was observed. I first iterated through all the cell files filtering out the cells whose 'divide' field is 1, which indicates a successful division. I stored the ID, birth frame and death frame of these cells in a structured format. Upon identifying the cells that have divided, the next step was to extract information about the frame numbers in which they had divided. The output 'clist' file is a multidimensional matrix that contains various information about all the cells throughout its life cycle. I extracted the 'age' field from the matrix to obtain the frame number in which the cell divided. I retrieved the matrix where each row corresponds to a cell ID and each column corresponds to the frame numbers from the clist matrix. The values in the matrix represent the age of the cell at each frame and when the cell divides the age sequence ends with a NaN value. The matrix looks like this: • Non-NaN values (numbers like 1, 2, 3, ...) indicate the cell ID in which it has been observed alive. • The transition from a number to a NaN implies that the cell has reached the end of its observable lifecycle in the dataset. This NaN can indicate either cell death or cell division. However, since we extracted only cells that had divided, the algorithm only checks for those cell IDs. For each cell ID that was determined to have divided (stored in divisionInfo.CellID), the frame number before the age turns to NaN is found and stored. This is done by matching CellIDs (the results of the previous step) to corresponding rows in the clist matrix. This method is used to retrieve the division frame as SuperSegger does not explicitly mark the division frame in the output files. After filtering the dataset for cells that show a division event, the number of transitions for divisions is counted. The probability of transition from state 0 (living) to state 2 (division) is calculated using the equation: …….. Eq: 9 Where: is the sum of the number of transitions from state 0 to state 2 for cells that have undergone division. is the sum of the number of transitions for all cells, regardless of the state transition. The probability of transition from the living state to the division state is given by the exponential distribution in the following equation. …….. Eq: 10 The division frequency kv is given by: …….. Eq: 11 Where: ln is the natural logarithmic function. Δt is the time interval between observations or frames. The process is automated through a code. The MATLAB script imports the transition count data from a designated CSV file, subsequently executing an analytical computation to determine the transition probability for cell division. Following this, it prompts the user for an experimental time interval to meticulously calculate the rate constant via logarithmic transformation and ratio analysis. Outlined below is a detailed summary of the procedures employed: Procedure Summary: 1. Load data - Purpose: Access pre-processed data needed for calculations. - Action: Load data like cell IDs, frame data, the center of mass and length of each cell. 2. Calculate Growth Rates - Purpose: Determine the rate at which each cell grows over time. - Instantaneous Growth Rate (IGR): - Action: Calculate the IGR using Equation 1. - Mean Growth Rate (MGR): - Action: Calculate the MGR using Equation 2. 3. Calculate Velocity - Purpose: Measure the movement speed and acceleration of cells. - Velocity: - Action: Compute the average 'r' of the center of mass and the geometric center using Equation 4. - Action: Compute displacement of 'r' between consecutive frames using Equation 5. - Action: Calculate velocity using Equation 3. 4. Determine Division Frequency - Purpose: Assess how frequently cells divide. - Action: - Calculate Total Transitions: Count all state transitions observed across all frames. - Calculate Division Transitions: Identify transitions for each cell from state 0 (living) to state 2 (division). - Calculate Division Probability: Determine the probability of transition from state 0 to state 2 using Equation 9. - Calculate Division Frequency: Compute the division frequency using Equation 11. 5. Compile and

store results - Purpose: Organize and save calculated data for further analysis. - Action: Save calculated values for growth rates, velocities, and division frequencies in a structured format for easy access, analysis, and future reference. 3.4.2 Results The anticipated outcomes of this experimental approach include a comprehensive set of sliding motility quantities for C. perfringens. The results offer a multidimensional view of the behavior of the cells through the quantification of the sliding motility by assessing growth patterns and the rate of cellular division. These quantities will provide valuable insights into the dynamic behavior of the bacteria over time. Growth Rate: The graph in Figure 25 illustrates the growth rates of three cells whose entire length is located within the field of view throughout the video, with cells 12 and 21 belonging to the same chain and cell 18 being part of the same colony but not the same chain (Fig. 28). Initially, both cells 12 and 21 exhibit higher growth rates that stabilize as they continue to elongate, reflecting the rapid initial growth phase typical of cells in a chain. Cell 18, while showing a similar trend, has more fluctuations in its growth rate, likely due to its different positioning within the colony and interactions with neighboring cells. A notable dip around 25 minutes is observed in all three cells, which is likely attributed to systemic technical artifacts such as frame drift. These observations suggest that the apparent growth rates can be affected by technical issues highlighting the necessity for corrections in frame drift calculations to obtain accurate growth measurements. Furthermore, the negative dips in the growth rate graphs are addressed in detail in subsequent paragraphs. Figure 25: The graph represents a comparative analysis of growth rates for three distinct fully seen cells over time. GR12, GR21 and GR18 represent the Growth Rates of cell ID 12, 21 and 18 respectively. One should note the artifacts when the growth rate is estimated for ‘half-seen’ cells that are partially outside the field of view in one or multiple frames of the video (Figs. 26 and 27). The cell length detected for these cells only reflects the length of the part that is observed in the field of view. One category of artifact growth rate trajectories (Fig. 26) demonstrates a pronounced initial growth rate that notably decreases over time. The initial rapid apparent growth is due to continuous movement into the field of view. The growth rate decreases over time and stabilizes to the true growth rate as the cells fully enter the frame for the following reasons. The ‘growth rate’ is estimated by Δ length / Δ time / current length. For ‘half-seen’ cells, Δ length / Δ time correlates with the speed of the cell. Since the speed of the cell is relatively constant over the course of the video, the estimated growth rate decreases as the "current length" increases. Once the cell is fully inside the field of view, the growth rate drops to the baseline that represents the actual growth rate. Cells 31 and 64 exhibit lateral contact with each other, indicating a close physical association as they move and grow within the same cluster. This connection likely influences their motility and growth behaviors, which are crucial for understanding their collective dynamics. In contrast, Cell 2 is part of a different group, isolated from the interactions observed between Cells 31 and 64 as shown in Figure 28. Figure 26: The graph represents a comparative analysis of growth rates for three half-seen distinct cells over time that follow a similar growth trend. GR2, GR31 and GR64 represent the Growth Rates of cell ID 2, 31 and 64 respectively. Another distinct category of half-seen cells (Fig. 27) is characterized by a stable growth rate with a sharp spike at 20 minutes. Cells 1, 3, and 4 are grouped closely together, forming a cohesive unit within the colony (Fig. 28). It is particularly intriguing to observe that the identified spikes in growth rates occur synchronously at precisely the 20-minute mark across multiple cells. Similar synchronous spikes also appear in Figures 25 and 26. These spikes are very likely artifacts resulting from technical processing. Biological factors are intrinsically noisy and unlikely to cause synchronous changes like this. The controlled experimental environment is unlikely to cause sharp transient spikes of this sort, either. Therefore, the most likely cause of the spikes is data processing artifacts, especially residual errors in frame drift correction. Frame drifts are caused by gradual or sudden movements of the sample with respect to the microscope lens.

Depending on the specific algorithm adopted by SuperSegger to correct these frame drifts, various levels of residual errors could emerge. Significant differences in the residual errors between consecutive frames would particularly contribute to systemic errors in the estimated key characteristics that rely on the calculation of differences between two consecutive frames, such as the growth rate in this part and the cell speed in the following section. These errors can hence lead to artifact synchronous spikes like those observed in Figures 25-27. Figure 27: The graph represents a comparative analysis of growth rates of three other half-seen cells over time that follow another unique growth trend. GR1, GR3 and GR4 represent the Growth Rates of cell ID 1, 3 and 4 respectively. Figure 28: The image showcases the cell IDs corresponding to the data represented in the graphs. Moreover, instances of desynchronized transient negative growth rates have been observed (Figs. 25, 26, 27), an anomaly considering that a reduction in cell length is biologically implausible. This occurrence is attributed to the cells altering their trajectories during movement, deviating from a straight path as shown in Figure 29. SuperSegger calculates cell length based on the linear distance between two points, not accounting for the actual curvilinear path taken by the cells. It is, therefore, necessary to acknowledge the limitations of the measurement techniques, which, when applied to cells exhibiting non-linear motion, could affect the accuracy of the growth rates. Additionally, errors in frame drift correction may also cause a negative spike, when, for instance, the errors in one frame cause an overestimate of the cell length and those in the following frame cause an underestimate. Nevertheless, these negative data only exist transiently for one separated time point. They are easy to recognize and discount. Figure 29: This image captures a few instances of non-linear cell movement, highlighted by the red circles. While only a select number of points are shown here, numerous similar instances occur throughout the video, showcasing frequent deviation. Velocity: While comprehensive data was obtained, for simplicity, the findings from a subset of cells are presented in Figure 30. The graph displays the speed of cells 20, 21, and 31 over time, each originating from different groups of cells. Each subplot showcases fluctuating speeds over time, suggesting that cells experience periods of varying activity. The behavior captured in the video shows cells moving rapidly for a brief period at certain time points. This is reflected in the speed profiles of the graph, indicating that the cells exhibit sudden bursts of speed at specific time points. It is striking to note the simultaneous occurrence of speed peaks across different cells. Because the displayed trajectories are from three cells in different groups, the synchronicity cannot be explained by correlation of their speeds due to physical interactions. Hence the most likely culprit is again the frame drift correction error. Note that particularly for ‘half-seen’ cells, frame drift correction may significantly affect the localization of their centers of mass, thus significantly shifting the estimated cell speed. It is striking to find many more spikes in the cell speed trajectories than the growth rate trajectories. This artifact needs to be analyzed more carefully in the follow-up work to pinpoint its cause and design a method of elimination. Figure 30: Graph showing the variability of speed in three distinct cells over time. S20, S21 and S30 represent the speeds of cell ID 20, 21 and 30 respectively. Division Frequency: Upon analyzing the data, the division frequency of the observed bacterial population is calculated. Over the 60-minute observation period, a total of 4 division transitions were recorded among the cells. The total number of transitions for all the states of the population is 4279. The time period between consecutive frames is approximately 30 seconds. Upon normalization with the total number of transitions, the division frequency kv is calculated as follows: By substituting the above value into the equation, we get …… Eq:11 ….. Eq: 12 The result signifies the rate at which the observed bacterial cells divided over the time course of the experiment, assuming the underlying process of cell division follows an exponential distribution. The derived division frequency offers a broad view of cellular kinetics and lays the foundation for understanding cell dynamics. 3.5 Discussion In this section, I will discuss the significance of my findings, explore

possible biological and technical explanations, address limitations, and suggest future research directions. The integration of SuperSegger with Omnipose proved highly effective for segmenting and tracking C. perfringens cells. This combination allowed for detailed analysis of cell growth rates. However, the transient negative growth rates observed in some instances highlight the need for more accurate methods to account for non-linear cell trajectories. During the automated analysis, cell length is calculated based on the linear distance between two terminal points of the cell. However, when cells curve or change orientation, this measurement erroneously interprets the movement as a decrease in cell length. To address this, a correction in the calculation method of cell length is necessary. Instead of relying on the linear distance between terminal points, adopting a curvilinear measurement that tracks the actual path of the cell's surface will provide more accurate and realistic data on cell growth rates. This approach will not only correct the transient negative growth rates but also enhance the overall accuracy of growth assessments in cells exhibiting non-linear movements. Incorporating more sophisticated algorithms that can handle complex cell movements will enhance the accuracy of future studies. Image analysis software that includes tools to trace the actual contour or periphery of a cell rather than just measuring the straight-line distance between two points can be used. Software like ImageJ can be explored for manual tracing of curved structures and could be adapted. Additionally, skeletonization can be used to represent the medial axis of the shape. This skeleton can then be used to calculate the true curvilinear length of the cell. Image processing libraries like OpenCV provide functions to perform skeletonization. One of the key observations from the analysis was the occurrence of spikes in cell growth rates and velocity at specific time points. The synchronous and transient nature of these spikes strongly suggest that they are artifacts caused by technical factors, particularly frame drift in the video. Frame drift refers to slight shifts in the imaging field between consecutive frames, which can occur due to vibrations or movement of the microscope stage. SuperSegger incorporates a frame alignment algorithm to perform frame drift correction, which adjusts for these shifts and aims to align the frames accurately. Despite this, residual errors in the frame drift correction still occur. These residual errors may not fully compensate for the drift, especially in cases of sudden or large movements. As a result, the residual error can cause artifacts, such as spikes in cell growth rates and velocity. It is these residual errors in the correction process that likely lead to the observed artifacts, rather than the initial uncorrected drift. Due to their significant sizes, these spikes could obscure important quantitative features of the true quantities, hindering true understanding the cellular dynamics. Improvement of the frame drift correction algorithm is hence important for accurate data acquisition. CHAPTER 4: CONCLUSION In conclusion, this thesis has effectively demonstrated the utilization of advanced image analysis techniques to analyze the growth kinetics and growth-driven sliding motility in C. perfringens cells. Initially, the LivecellX image segmentation method was employed; however, it failed to provide the necessary results and accuracy required for detailed analysis. This challenge prompted a shift to SuperSegger, which proved to be highly effective. With SuperSegger, I was able to achieve precise segmentation of the cells. This enhanced segmentation enabled the use of computational techniques to accurately track the sliding motility of the bacteria. The findings from this study underscore the capabilities of SuperSegger in complex biological imaging. The integration of the Omnipose model within SuperSegger 2, which was trained on datasets similar to the dataset used in my research was instrumental in realizing this superior performance. This research highlights the importance of selecting appropriate tools for image analysis in microbiological studies. Subsequently, the outputs of SuperSegger enabled the successful calculation of key characteristics associated with the cells' sliding motility, including growth rates, division frequencies, and velocities. The work demonstrates the use of suitable techniques in extracting essential characteristics of motility behavior

and sets a foundation for future work in this area to enhance our understanding of microbial behaviors. These metrics have provided new insights into the dynamic behaviors of C. perfringens. Moving forward, this research sets the stage for several promising directions. Future work could look deeper into improving the accuracy of frame drift correction. Residual errors in frame drift correction can lead to significant artifacts in the data, such as the observed spikes in growth rates and velocity. Therefore, future research should aim to refine the algorithms used for frame drift correction to minimize these errors. This could involve developing more sophisticated methods for detecting and compensating for the drift or incorporating advanced computational techniques to better handle residual errors. One possible solution can be to implement iterative correction methods that repeatedly adjust the alignment based on detected residual errors. Each iteration can refine the accuracy of the alignment. That being said, minimizing drift during the imaging itself can be the best way to eliminate these issues altogether. Improvements in imaging hardware, such as more stable microscope stages and higher precision cameras, can reduce the likelihood of frame drift. Continuous calibration and maintenance of the equipment, along with controlled environmental conditions to minimize vibrations, can also help mitigate these issues. Therefore, the best approach for future studies is to obtain videos without significant frame drift. Another avenue for future research is improving the measurement techniques for cell motility, particularly in capturing the non-linear paths cells travel. Non-linear paths, where cells deviate from straight trajectories, often reflect more naturalistic interactions within a cellular environment. Particularly, SuperSegger is being regularly updated with new features and refining existing functionalities. Therefore, it might be advantageous to communicate the challenges with their team. Specifically addressing the issues with tracking non-linear cell movements could prompt valuable enhancements in future versions of the software. An alternative would be to explore additional advanced imaging algorithms that can more accurately trace these curved paths and enhance the precision of growth rate calculations. In conclusion, this research project not only enables collection of data that will enhance our understanding of C. perfringens' dynamic behaviors but also point out needed improvement in the AI-based image segmentation and data post-processing. Appendix Code 1: To Calculate IGR and MGR % Assuming the data is stored in a table named 'data' with columns 'Time' and 'cellLength' Time = data.Time; cellLength = data.cellLength; % Calculate Instantaneous Growth Rates (IGR) % Ensure calculations are done between successive pairs of time points and lengths IGR =

zeros(length(Time)-1, 1); for i = 1:length(Time)-1 IGR(**i**) = (cellLength(**i** +1) - cellLength(**i)) / (Time(i+1) - Time(i**

)); end % Calculate Mean Growth Rate (MGR) MGR = (cellLength(end) - cellLength(1)) / (Time(end) - Time(1)); Code 2: To Calculate Velocity and Acceleration % Loop through each file found in the folder for i = 1:length(files) filename = fullfile(folderPath, files(i).name); data = load(filename); % Load the data from the file % Extract position data for r_center, rcm, and time r_center = data.r_center; % Assuming r_center is stored as an Nx2 matrix [x, y] rcm = data.rcm; % Assuming rcm is stored as an Nx2 matrix [x, y] time = data.time; % Assuming time is a column vector of length N % Calculate average positions avg_position = (r_center + rcm) / 2; % Nx2 matrix [x_avg, y_avg] % Initialize displacement, velocity, and acceleration arrays for this file displacements = zeros(size(avg_position, 1) - 1, 1); velocities = zeros(size(avg_position, 1) - 1, 1); accelerations = zeros(size(avg_position, 1) - 2, 1); % Calculate displacement and velocity for each frame for j = 1:size(avg_position,

1) - 1 % Calculate time interval between the current frame and the next delta_t = time(j+1) - time(j); % Time interval between frames % Displacement displacement = sqrt((avg_position(

j+1, 1) - avg_position(j, 1))^2 + ... (avg _position(j+1 , 2) - avg _position(j , 2))^2); displacements(j 43

) = displacement; % Velocity if delta_t != 0 % To avoid division by zero velocity = displacement / delta_t; else velocity = 0; % Assign zero velocity if time difference is zero end velocities(j) = velocity; end end % Calculate acceleration for each velocity transition for k = 1:size(velocities, 1) - 1 time_interval = time(k+2) - time(k+1); % Interval between velocity measurements if time_interval != 0 acceleration = (velocities(k+1) - velocities(k)) / time_interval; else acceleration = 0; % Assign zero acceleration if time difference is zero end accelerations(k) = acceleration; end % Store or display results all_velocities = [all_velocities; velocities]; % Append current velocities to overall list all_accelerations = [all_accelerations; accelerations]; % Append current accelerations to overall list % Optionally, save results to a file save(fullfile(folderPath, sprintf('velocity_acceleration_results_%s', files(i).name)), 'velocities', 'accelerations'); end Code 3: To calculate the division frequency. % Initialize the table with appropriate variable names divisionInfo = table([], [], [], 'VariableNames', {'cellID', 'MotherID', 'DaughterIDs'}); % cellFiles = dir(fullfile(cellFileFolder, 'cell*.mat')); % Assuming cellFileFolder is already defined for i = 1:length(cellFiles) % Load the current cell's data currentCellData = load(fullfile(cellFileFolder, cellFiles(i).name)); % Check if the cell has divided if currentCellData.divide == 1 % Extract and store the required information cellID = currentCellData.ID; % Assuming the ID field exists motherID = currentCellData.motherID; % Assuming the motherID field exists % Assuming daughterID field exists and contains IDs of daughter cells % Here, we convert the IDs to a comma-separated string for easy Excel storage daughterIDsStr = strjoin(arrayfun(@num2str, currentCellData.daughterID, 'UniformOutput', false), ','); % Add the information to the table newRow = {cellID, motherID, daughterIDsStr}; divisionInfo = [divisionInfo; newRow]; end end IDs = squeeze(clist.data3D(:,1,:)); % initialize a vector to store division frames divisionFrames = zeros(height(divisionInfo), 1); for i = 1:height(divisionInfo) cellID = divisionInfo.CellID(i); % Extract the age sequence for the current cell ID ageSequence = IDs(cellID, :); % Find the first NaN value in the age sequence which indicates division divisionFrame = find(isnan(ageSequence), 1, 'first'); % Store the division frame divisionFrames(i) = divisionFrame; end % Add the division frames to the divisionInfo table divisionInfo.DivisionFrame = divisionFrames; % Get a list of all '.mat' files in the directory files = dir(fullfile(directory, 'cell*.mat')); transitionsData = []; % Initialize the array to hold transition data % Loop through each file in the directory for i = 1:length(files) % Load data from the .mat file data = load(fullfile(directory, files(i).name)); % Check if both 'birth' and 'death' fields are present in the data if isfield(data, 'birth') && isfield(data, 'death') birth_frame = data.birth; % Extract the birth frame death_frame = data.death; % Extract the death frame % Calculate the number of transitions for the current cell transitions = death_frame - birth_frame; % Store the data transitionsData = [transitionsData; {files(i).name, birth_frame, death_frame, transitions}]; end % Optionally, print each cell's transition details fprintf('%s: Birth = %d, Death = %d, Transitions = %d\n', ... files(i).name, birth_frame, death_frame, transitions); end end % Read data from CSV file data = csvread('transitions.csv', 1, 0); % Extract ni_0_to_2 and ni from the data ni_0_to_2 = data(:, 1); % Assuming this is the first column ni = data(:, 2); % Assuming this is the second column % Calculate the

probability P(0 -> 2) P_0_to_2 = sum(ni_0_to_2) / sum(ni); % Ask for the time interval Δt % User is prompted to enter the value for delta_t delta_t = input('Enter the value for Δt: '); % Compute 1 - P(0 -> 2) to find the complement of the probability one_minus_P = 1 - P_0_to_2; % Calculate the natural logarithm of 1 - P(0 -> 2) ln_one_minus_P = log(one_minus_P); % Divide by Δt to find kV kV = ln_one_minus_P / delta_t; % Display the results fprintf('The probability P(0 -> 2) is: %f\n', P_0_to_2); fprintf('The calculated value of kV is: %f\n', kV); Code 4: To link the frames after segmentation to match cell IDs

```
import pandas as pd def read_data(file_path): return pd.read_csv(file_path) def update_ids( data
```

11

, start_frame, end_

```
frame): for frame in range(start_frame, end_frame
```

23

, -10): # Update backward for overlaps if frame > 10: next_frame = frame - 10 update_cell_ids(data, frame, next_frame) if start_frame == 111: update_cell_ids_forward(data, start_frame, 121) def update_cell_ids(data, current_frame, next_frame): for idx, row in data.iterrows(): current_bbox = row[f'BoundingBoxFrame{current_frame}N'] # Find matching bounding box in the next_frame match = data[data[f'BoundingBoxFrame{next_frame}L'] == current_bbox] if not match.empty: new_id = row[f'CellIDFrame{current_frame}N'] old_id = match[f'CellIDFrame{next_frame}L'].values[0] data.loc[data[f'CellIDFrame{next_frame}'] == old_id, f'CellIDFrame{next_frame}'] = new_id def update_cell_ids_forward(data, start_frame, end_

```
frame): for frame in range(start_frame, end_frame + 1 ): next_ frame = frame + 1 if next_ frame
```

23

<= end_frame: data[f'CellIDFrame{next_frame}'] = data[f'CellIDFrame{frame}']

```
def save_data(data, file_path): data.to_csv(file_path, index=False
```

11

) data_file_path = 'data.csv' output_file_path = 'updated_data.csv' data = read_data(data_file_path) update_ids(data, 111, 11) save_data(data, output_file_path)

References Journals:

Carpenter, A. E., Jones, T. R., Lamprecht, M. R., Clarke, C., Kang, I. H., Friman, O., Guertin, D . A., Chang, J. H., Lindquist, R. A., Moffat, J., Golland, P., & Sabatini, D. M. (2006). CellProfiler: image analysis software for identifying and quantifying cell phenotypes. Genome Biol, 7(10), R100. https://doi.org/10.1186/gb-2006-7-10-r100 **Cutler** 27

,

K. J., Stringer, C., Lo, T. W., Rappez, L., Stroustrup, N., Brook Peterson, S., Wiggins, P 34

. A., & Mougous, J. D. (2022). Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation. Nat Methods, 19(11), 1438-1448. https://doi.org/10.1038/s41592-022-01639-4

Holscher, T., & Kovacs, A. T. (2017). Sliding on the surface: bacterial spreading without an active motor. Environ Microbiol, 19 (7), 2537-2545. **https://doi.org/10.1111/1462- 2920.13741** Jakimovski, G., & Davcev, **D** 13

. (

2018). Lung cancer medical image recognition using Deep Neural Networks . 2018 **Thirteenth International Conference on Digital Information Management (ICDIM** 29

),

Liu, H., Bouillaut, L., Sonenshein , A. L., & **Melville** , S. B. (**2013). Use of a mariner-based transposon mutagenesis system to isolate Clostridium perfringens mutants deficient in gliding motility. J Bacteriol, 195** (3), 629-636. **https://doi.org/10.1128/JB.01288-12 Liu** 9

,

H., McCord, K. D., Howarth, J., Popham, D. L., Jensen, R. V., & Melville, S. B. (2014 33

). Hypermotility in Clostridium perfringens strain SM101 is due to spontaneous mutations in genes linked to cell division. J Bacteriol, 196(13), 2405-2412. https://doi.org/10.1128/JB.01614-14

McMahon, S. G., Melville, S. B., & Chen, J . (2022). **Mechanical limitation of bacterial motility mediated by growing cell chains. Biophysical Journal, 121(12), 2461-2473** 3

.

Mehdizadeh Gohari, I., M, A. N., Li, J., Shrestha, A., Uzal, F., & B, A. M. (2021). Pathogenicity and virulence of Clostridium perfringens. Virulence, 12(1), 723-753. https://doi.org/10.1080/21505594.2021.1886777 1

Padovani, F., Mairhormann, B., Falter-Braun, P., Lengefeld, J., & Schmoller, K. M. (2022). Segmentation, tracking and cell cycle analysis of live-cell imaging data with Cell-ACDC. BMC Biol, **20(1), 174. https://doi.org/10.1186/s12915-022-01372-6** 8

Powell, E. O. (1956). Growth rate and generation time of bacteria, with special reference to continuous culture. J Gen Microbiol, 15(3), 492-511. https://doi.org/10.1099/00221287- 15-3-492 10

Russell, S. J., & Norvig, P. (2016). Artificial intelligence: a modern approach. Pearson 14

.

Ryan, K. J., Ray, C. G., & Sherris, J. C. (2004). Sherris medical microbiology: an introduction to infectious diseases 30

. (No Title).

Stringer, C., Wang, T., Michaelos, M., & Pachitariu, M. (2021). Cellpose: a generalist algorithm for cellular segmentation . Nat **Methods, 18** (1), 100-106. **https://doi.org/10.1038/s41592- 020-01018-x Stylianidou, S., Brennan, C., Nissen** , S. B., **Kuwada** , N. J., & **Wiggins** , P. A. (**2016). SuperSegger: robust image segmentation, analysis and lineage tracking of bacterial cells** . Mol Microbiol, **102** (4), 690-700. **https://doi.org/10.1111/mmi.13486** 6

Varga, J. J.,

Nguyen, V., O'Brien , D. K., **Rodgers, K., Walker** , R. A., & **Melville** , S. B. (**2006**). Type IV pili-dependent gliding motility in the Gram-positive pathogen Clostridium perfringens and other Clostridia. Mol Microbiol, 62(3), 680-694. https://doi.org/10.1111/j.1365-2958.2006.05414.x 7

Varga , J. J., **Therit, B., & Melville** , S. B. (**2008). Type IV pili and the CcpA protein are needed for maximal biofilm formation by the gram-positive anaerobic pathogen Clostridium perfringens. Infect Immun, 76** (11), 4944-4951. **https://doi.org/10.1128/IAI.00692-08**

Wang, W., Taft, D. A., Chen, Y. J., Zhang, J., Wallace, C. T., Xu, M., Watkins, S. C., & Xing, J. (2019). Learn to segment single cells with deep distance estimator and deep cell detector. Comput Biol Med, 108, 133-141. https://doi.org/10.1016/j.compbiomed.2019.04.006 4

Articles: Aryal, Sagar. (2023). Clostridium perfringens - An Overview. MicrobeNotes, Retrieved from https://microbenotes.com/clostridium-perfringens/ Biologydictionary.net Editors. (2017). Microbiology. Retrieved from https://biologydictionary.net/microbiology/

Centers for Disease Control and Prevention. (2018). Foodborne illnesses and germs. Retrieved from https://www.cdc.gov/foodsafety/foodborne-germs.html 24

Rancho Labs. (2021). Supervised Learning Algorithm. Medium. Retrieved from https://rancholabs.medium.com/supervised-learning-algorithm-b5d109f89e8d Piyush, & Rishabh. (2020). 3

types of machine learning. Retrieved from https://www.newtechdojo.com/3-types-of-machine-learning 12

/#google_vignette Tyagi, M. (2023). Image segmentation explained; Built In. Retrieved from https://builtin.com/machine-learning/image-segmentation Yadav, R. (2024). Basic image processing in machine learning; Medium. Retrieved from https://medium.com/p/ae5115659844 Images: Centers for Disease Control and Prevention (1974). Gram-positive, Clostridium perfringens bacteria.

Public Health Image Library (PHIL). Retrieved from https://phil.cdc.gov/details.aspx?pid 40

=2995 Klingler, N. (2024). Segmentation using deep learning. Viso AI. Retrieved from

https://viso.ai/deep-learning/image-segmentation-using-deep-learning 50

/ Piyush, & Rishabh. (2020). 3

types of machine learning. Retrieved from https://www.newtechdojo.com/3-types-of-machine-learning 12

/#google_vignette Raj, R. (2024).

Classification and Regression in Machine Learning . EnjoyAlgorithms. Retrieved from **https** ://www.enjoyalgorithms. **com** /blogs/ **classification** -and- **regression-in- machine-learning** 21

/

Rajbanshi, S . (2021). **Everything you need to know about machine learning** . Analytics **Vidhya** . Retrieved from **https://www.analyticsvidhya.com/blog/2021/03/everything-you-need-to- know-about-machine-learning** 17

/ Rancho Labs. (2021). Supervised Learning Algorithm. Medium. Retrieved from https://rancholabs.medium.com/supervised-learning-algorithm-b5d109f89e8d

Rancho Labs. (2021). Unsupervised Learning Algorithm. Medium. Retrieved from https://rancholabs.medium.com/unsupervised-machine-learning-13ada997b2bd

Singh, S. (2018). Cousins of Artificial Intelligence:

Artificial intelligence, Machine Learning and **Deep Learning. Towards Data Science. Retrieved from https://towardsdatascience.com** 14 /cousins-of- **artificial-intelligence**

-dda4edc27b55 Wiggins Lab. (2017). SuperSegger-cell data files, GitHub. Retrieved from https://github.com/wiggins-lab/SuperSegger/wiki/The-cell-data-files Wiggins Lab. (2024). SuperSegger modified for Omnipose. GitHub. Retrieved from https://github.com/tlo-bot/SuperSegger-omnipose 1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59

sources:

1 35 words / < 1% match - from 05-Aug-2023 12:00AM
vtechworks.lib.vt.edu

2 30 words / < 1% match - Internet from 27-Sep-2022 12:00AM
vtechworks.lib.vt.edu

3 24 words / < 1% match - Internet from 13-Jan-2023 12:00AM
vtechworks.lib.vt.edu

4 50 words / < 1% match - Internet from 05-Aug-2022 12:00AM
www.biorxiv.org

5 18 words / < 1% match - from 06-Aug-2023 12:00AM
www.biorxiv.org

6 54 words / < 1% match - from 27-Apr-2024 12:00AM
elifesciences.org

7 46 words / < 1% match - Internet from 10-Aug-2022 12:00AM
docksci.com

8 40 words / < 1% match - from 07-Aug-2023 12:00AM
era.ed.ac.uk

9 36 words / < 1% match - from 10-Jan-2024 12:00AM
ebin.pub

10 33 words / < 1% match - Internet from 25-Oct-2019 12:00AM
www.jmbfs.org

11 30 words / < 1% match - from 22-Apr-2024 12:00AM
www.codewithc.com

12 30 words / < 1% match - Internet from 15-Oct-2022 12:00AM
www.singhealthacademy.edu.sg

13 28 words / < 1% match - Crossref Posted Content
Yongjin Liu, Sandra LaBonte, Courtney Brake, Carol LaFayette, Adam P. Rosebrock, Amy A. Caudy, Paul D. Straight. "MOB rules: Antibiotic Exposure Reprograms Metabolism to Mobilize Bacillus subtilis in Competitive Interactions", eLife Sciences Publications, Ltd, 2024

14 28 words / < 1% match - Internet from 30-Oct-2020 12:00AM
journalmodernpm.com

15 17 words / < 1% match - from 31-Dec-2023 12:00AM
fastercapital.com

16 9 words / < 1% match - from 20-Feb-2024 12:00AM
fastercapital.com

17 26 words / < 1% match - from 22-Apr-2023 12:00AM
ouci.dntb.gov.ua

18	17 words / < 1% match - Internet from 07-Nov-2022 12:00AM www.nature.com
19	9 words / < 1% match - Internet from 13-Nov-2019 12:00AM www.nature.com
20	25 words / < 1% match - from 25-Dec-2023 12:00AM linorg.usp.br
21	23 words / < 1% match - ProQuest Lakhanpal, Pranshul. "Balancing Privacy and Accuracy in IoT Using Domain-Specific Features for Time Series Classification", California Polytechnic State University, 2023
22	22 words / < 1% match - Crossref "Proceedings of the 8th International Conference on Computational Science and Technology", Springer Science and Business Media LLC, 2022
23	22 words / < 1% match - from 02-May-2024 12:00AM blenderartists.org
24	21 words / < 1% match - Internet from 16-Jan-2023 12:00AM ufdcimages.uflib.ufl.edu
25	20 words / < 1% match - Internet from 25-Jul-2020 12:00AM curis.ku.dk
26	20 words / < 1% match - Internet from 16-Dec-2022 12:00AM www.foodsafetynews.com
27	19 words / < 1% match - from 28-Mar-2024 12:00AM refubium.fu-berlin.de

28 19 words / < 1% match - Internet from 24-Jun-2021 12:00AM
uknowledge.uky.edu

29 18 words / < 1% match - from 27-May-2023 12:00AM
link.springer.com

30 18 words / < 1% match - Internet from 17-Jan-2023 12:00AM
yorkspace.library.yorku.ca

31 17 words / < 1% match - from 25-Dec-2023 12:00AM
www.surfactants.net

32 16 words / < 1% match - Internet from 14-Jan-2023 12:00AM
medium.com

33 16 words / < 1% match - Internet from 28-Aug-2022 12:00AM
www.intelligent-imaging.com

34 16 words / < 1% match - from 13-Apr-2024 12:00AM
www2.mdpi.com

35 15 words / < 1% match - Crossref
Deepshikha Ghosh, Anushka Biswas, Mithun Radhakrishna. "Advanced computational approaches to understand protein aggregation", Biophysics Reviews, 2024

36 15 words / < 1% match - ProQuest
Patel, Tirth Nimishbhai. "Investigating the Role of Vasopressin Receptor 1a Expressing Neurons in Mouse Dorsal Raphe", Rowan University, 2023

37 14 words / < 1% match - Crossref

Khajamoinuddin Syed, William C. Sleeman, Joseph J. Nalluri, Rishabh Kapoor, Michael Hagan, Jatinder Palta, Preetam Ghosh. "Artificial intelligence methods in computer-aided diagnostic tools and decision support analytics for clinical informatics", Elsevier BV, 2020

38 14 words / < 1% match - ProQuest
Li, Kaixuan. "Real-Time Image Analysis-Based Automatic Optic Disc Recognition and Blood Vessels Detection in Fundoscope Images", Virginia State University, 2023

39 14 words / < 1% match - from 21-Mar-2024 12:00AM
www.coursehero.com

40 14 words / < 1% match - from 12-Apr-2023 12:00AM
www.researchgate.net

41 12 words / < 1% match - from 20-Jan-2024 12:00AM
corpus.ulaval.ca

42 12 words / < 1% match - Internet from 15-Oct-2022 12:00AM
nebula.wsimg.com

43 11 words / < 1% match - Internet
Iqbal, Usman Zafar. "MPD: Simulation and Control Systems", University of Stavanger, Norway, 2011

44 11 words / < 1% match - Internet from 17-Oct-2022 12:00AM
scian.cl

45 10 words / < 1% match - Crossref
Goran Jakimovski, Danco Davcev. "Lung cancer medical image recognition using Deep Neural Networks", 2018 Thirteenth International Conference on Digital Information Management (ICDIM), 2018

46 10 words / < 1% match - Crossref Posted Content
Mario Vitacolonna, Roman Bruch, Ane Agaçi, Elina Nürnberg et al. "A Multiparametric Analysis Reveals Differential Behavior of Spheroid Cultures on Distinct Ultra-Low Attachment Plates Types", Cold Spring Harbor Laboratory, 2024

47 10 words / < 1% match - Internet from 29-Oct-2022 12:00AM
chen.biol.vt.edu

48 10 words / < 1% match - Internet
Rand, Kristina Marie. "Doctor of Philosophy", University of Utah, 2014

49 10 words / < 1% match - Internet from 02-Jul-2020 12:00AM
www.nbi.ku.dk

50 10 words / < 1% match - Internet from 17-Mar-2022 12:00AM
www.openj-gate.com

51 9 words / < 1% match - Crossref
"Advanced Concepts for Intelligent Vision Systems", Springer Science and Business Media LLC, 2023

52 9 words / < 1% match - Crossref
Mahad S. Baawain, Aisha S. Al-Serihi. "Systematic Approach for the Prediction of Ground-Level Air Pollution (around an Industrial Port) Using an Artificial Neural Network", Aerosol and Air Quality Research, 2014

53 9 words / < 1% match - ProQuest
Moradi, Fereshteh. "Application of a Deep Learning Approach to Map and Predict the Age-Friendliness of the Built Environment", University of Technology Sydney (Australia), 2023

54 9 words / < 1% match - Crossref
Zhijiang Wan, Manyu Li, Zihan Wang, Hai Tan, Wei Li, Lisu Yu, Dinesh Jackson Samuel. "CellT-Net: A Composite Transformer Method for 2-D Cell Instance Segmentation", IEEE Journal of Biomedical and Health Informatics, 2023

55 9 words / < 1% match - Internet from 28-Feb-2023 12:00AM
www.mcours.net