ABSTRACT

JAMESON, CARTER D. Data Science and Mathematical Modeling Approaches to Studying Collective Motion. (Under the direction of Kevin Flores.)

The mathematics behind collective motion remains an open area of research, as these intriguing population dynamics were only recently formalized using methods from statistical mechanics. Significant obstacles include the incorporation of both heterogeneous and collective behaviors, and the unique range of physical scales required to understand collective motion. Because collective motion is relevant to researchers in many fields including sociology, particle physics, and cell biology, there are many approaches to studying these open questions. Taking an applied mathematics and data science approach, this dissertation aims to advance the current understanding of collective motion by applying computer vision, agent-based modeling, and topological data analysis techniques to both simulated particle motion data and a dataset of time-lapse microscopy images of fibroblast cell nuclei collected in the course of a scratch assay study.

While state-of-the-art computer vision techniques based on deep learning provide promising methods for examining the collective motion behavior of new microscopy images, we are often impeded by the requirement of large datasets of labeled training data. We address this limitation by introducing a robust convolutional neural network for cell nucleus detection. Concrete results on our experimental data show that the detector can infer accurate locations on unseen images after being trained from scratch using one annotated image from a single video. By applying this approach to a novel time-series dataset of experimental scratch assay images, we obtain detections that outperform computer vision methods that do not leverage deep learning. We use these nucleus detections to obtain trajectories of the cells through time using the Crocker-Grier particle tracking algorithm. Our subsequent analysis of these trajectories includes using an agent-based stochastic differential equation formulation of Brownian motion to model the random walk behavior of the cells, and the use of topological data analysis to distinguish between confluent and over-confluent cell populations.

In addition to these conclusions involving real-world data, we present a novel parameter estimation framework based on the topology or density of simulated collective motion data. We simulate prototype point-cloud datasets using a stochastic variant of the agent-based D'Orsogna model of interactive particle motion. Using least-squares optimization on summaries of the topology and particle density, we show that it is feasible to recover the model parameters used to generate these datasets. This approach, based on standard

Nelder-Mead direct search optimization, is able to estimate the parameters of deterministic and stochastic variants of our candidate model. By expressing this parameter estimation task as an inverse problem, we can approximate distributions of model parameter estimates, and are thus able to directly compare the efficacy of different methods.

Data Science and Mathematical Modeling Approaches to Studying Collective Motion

by
Carter D. Jameson

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2021

APPROVED BY:

Jason Haugh	Ralph Smith
Hien Tran	Kevin Flores Chair of Advisory Committee

DEDICATION

To my wife, Anna, for marrying me during graduate school.

BIOGRAPHY

The author was born in North Kansas City, Missouri and lived in the Show-Me State until moving to Oklahoma for college. He attended The University of Tulsa from 2013 to 2017, obtaining a B.A. in Music, a B.S. in Mathematics, and an M.S. in Applied Mathematics. He moved to Raleigh, North Carolina in 2017 to pursue a PhD in Applied Mathematics, and discovered the joy of computational math and programming. He now resides in Albuquerque, NM with his wife, Anna Jameson, and looks forward to his upcoming career as a computer scientist at Sandia National Labs.

ACKNOWLEDGEMENTS

I have to start by thanking my advisor, Dr. Kevin Flores, for being exactly the kind of mentor I needed as a doctoral student. He was always willing to let me be a self-starter, pursuing the ideas that interested me, but always gave me direction and plans when I needed to make real progress. I am similarly indebted to my entire research group, especially Dr. John Lagergren. I frankly would not have been aware of half the projects I worked on over the past few years without his wealth of knowledge, and he was always happy to offer an exciting idea. Most crucial to this dissertation work is Dr. John Nardini, post-doc extraordinaire. Not only was his background knowledge in both scratch assay modeling and topological data analysis the basis of this entire dissertation, but he put hundreds of hours into reviewing my work and meeting with me, all while on the tireless academia job hunt. John has been the non-faculty mentor most people only dream of collaborating with, and I hope we can find ways to work together in the future. My committee has been duly helpful in directing my research; much of my deep learning knowledge comes from Dr. Hien Tran's lectures, Dr. Ralph Smith provided indispensable guidance as an expert in parameter estimation, and Dr. Jason Haugh helped me understand the requisite cell biology. Similarly, I would not have done any of this research without Scott Baldwin, who provided the dataset that sparked this investigation, and who made sure I was not completely clueless when it came to describing the scratch assay experiments.

I have to thank friends at NC State who helped me learn how to code. As someone who came into graduate school with minimal practical programming experience, I could not have done all these computational experiments without the guidance of Stephen Gilmore, Robin Morillo, and my research group colleagues, John Nardini, John Lagergren, and Dr. Erica Rutter. I am so grateful to the friends I made in the mathematics department, especially since I moved to North Carolina with no acquaintances in the area. To Tyler Knapp, Elisabeth Congdon, Anthony Powell, Jane Coons, Zev Woodstock, Kate Pearce, Isaac Sunseri, Robin, John, Stephen, and many others; I would have been lost without you those first couple of years, and I hope our paths cross again.

I would not, however, have ever made it here without the selfless support of my family. To Mom, Dad, Pop, and Granny—I did it! To my parents, thanks for giving this geeky kid every resource he needed to learn everything he needed to about mathematics, music, computers, and everything in between. To Mom, I would have never have found the school I wanted to attend without your penchant for research. Most of all thank you for the love and for the many, many late nights of writing papers and doing math homework. The

rest of my family has sacrificed a lot over the course of my education, with my siblings Parker, Emme, and Ella, helping me move to another state and dealing with extended periods of my being too busy. My in-laws have also been indispensably supportive over the course of the Covid-19 pandemic, when they welcomed us into their homes and been the only friends we have in town. To Ron, thanks for helping me get the internship, not only did it help me make some money to pay off the ring I bought your daughter, but it turned into a career. To Susan, thanks for welcoming me into your family and your home, and for feeding me well over these past few months of intense writing.

Finally, of course, I could never thank my wife enough for what she has given me over the past three years. Anna, you dealt with the 1800 miles between us with grace, and then moved all that way just for me, which is nothing short of a miracle in my mind. Saying yes to marrying me is the best gift I have ever received, and I thank God for you every day. However, taking care of everything while I finish my degree and continuing to be patient with my education is a close second. You have been our breadwinner, and you have dealt with more than your fair share of late nights. A long treatise on applied mathematics might not have been the repayment you had in mind, but nevertheless, this is truly all for you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 The Mathematics of Data Science

The study of applied mathematics has become vastly more intriguing as computing tools and powerful mathematical software have achieved ubiquity in recent decades. Of particular interest to the modern applied mathematician is the development of algorithms and computational methods for the sake of understanding the data that are increasingly available to researchers in nearly every scientific discipline. This interest is one of the general aims of the contemporary field of *data science*. Prominent computer scientists David Blei and Padhraic Smyth describe data science as "the child of statistics and computer science" [1], yet—just as both of these disciplines are ostensibly direct descendants of mathematics—many data science principles are established using mathematics. While many data science techniques are based on explicitly-statistical methods like regression, classification, decision trees, and clustering, strict mathematical reasoning underlies many other methods.

In particular, the development of new and novel data science tools often requires an in-depth knowledge of mathematics. The neural networks used in deep learning are a perfect example of this demand. Deep learning has quickly become a state-of-the-art solution for many data science tasks, particularly as a popular tool which is available to any computer-savvy developer, even those without formal training in mathematics. As the programming interfaces for these technologies become more prevalent and user-friendly, deep learning practitioners become further removed from the mathematical foundation on which these tools are built. In reality, training a neural network involves employing complicated matrix calculus in the form of backpropagation. The ability to even take derivatives of network outputs with respect to individual parameters is also based on the continuity and differentiability of nonlinear internal activation functions [2].

Furthermore, the methods that best use these gradients are based on optimization theory, which may leverage decades of rigorous theorems from game theory, functional analysis, and control theory [3]. While there is certainly no fault in researchers using tools to their fullest extent in search of scientific truths and the significance of novel data, we rely on those who understand the mathematics of matrices, high-dimensional spaces, functions, operators, and their applications to imagery, time series, and categorical data to design and implement these tools.

We appeal to this sense of significance in justifying the following research. In aiming to develop practical methods for understanding collective motion, we invoke computational geometry, algebraic topology, probability theory, statistical learning, information theory, matrix theory, functional analysis, optimization, calculus, and floating-point arithmetic. While a exhaustive understanding of each of these fields and subfields is not required to understand our work, or even to perform the enclosed investigations, a knowledge of how these fields tend to view problems and how they work together is essential for creating a tool that can glean useful, meaningful conclusions from scientific data. Altogether, we attempt to introduce rigor into data-scientific methods whenever possible, while also endeavoring to remain close to the application of interpreting collective motion for the sake of proposing ideas that are relevant to scientists in any field of study.

1.2 Background

This work consists of multiple distinct efforts which are unified by the comprehension of *collective motion*. Our research investigates experimental cell microscopy data and simulated particle data that exhibit *emergent behaviors*, and pursues a novel understanding of collective biological motion. We do this using *agent-based models*, which consider individuals as the unit of interest in a population. This allows us to represent the data as a *point cloud*, enabling the use of tools which describe collective motion.

1.2.1 Collective Motion

Practical engineering has long perceived the propensity of humans and living things in orderly, coherent groups. From the so-called "Magic Roundabout" installed in England in the 1970's [4], to Craig Reynolds' Boids software [5], inventors have understood the ability to control heterogeneous populations with simple rules for decades. The underlying physical causes of these behaviors and how these causes relate to one another on a variety of scales is, however, a more recent discovery.

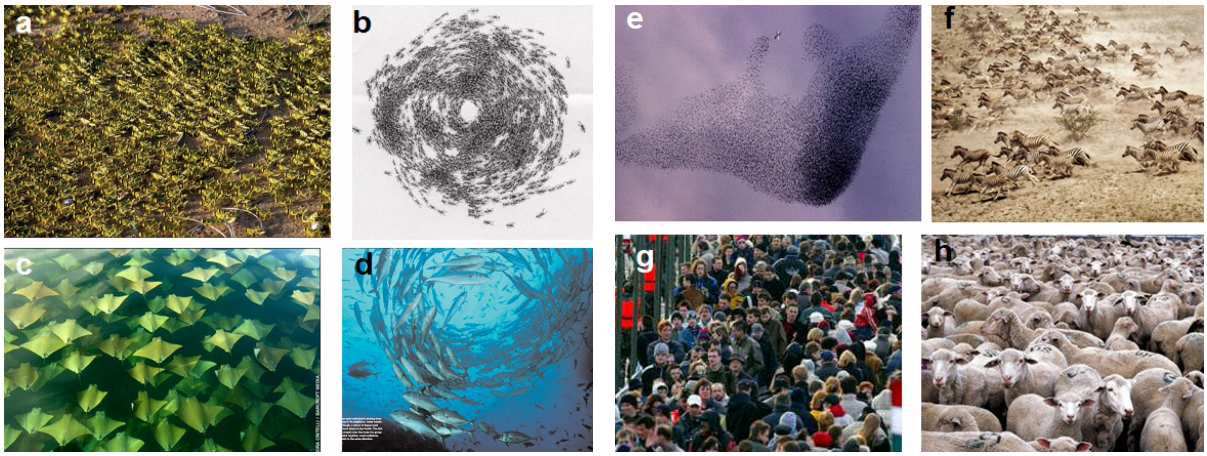

Figure 1.1: Images showing collective motion and flocking patterns in groups of organisms. Adapted from Vicsek and Zafeiris [6].

The scientific study of collective motion originates in physics and the study of the statistical mechanics of phase transitions [7]. These tools were adopted by Tamás Vicsek, who proposed that the eye-catching patterns in flocks of birds and schools of fish could be described using this same statistical mechanics framework based on the notion that these organisms also move primarily due to the forces of self-propulsion and interaction. To demonstrate this, he first showed—using a simple particle motion model—that organisms who seek order do in fact become ordered given a finite amount of disturbance [8]. Over more than a decade of research, Vicsek codified many of the notions that we take for granted in the modern study of collective motion, including the application of the laws of thermodynamics to biological systems and the connection between collective motion and *emergence* [6].

Emergence describes the appearance of macro-scale dynamics as a consequence of micro-scale self-organization, [9] and in particular describes how these patterns arise from heterogeneous actions within a population [10]. Vicsek describes how these emergent behaviors are responsible for the complex patterns present in these biological collective motion systems, and furthermore how they can be brought about with conceptually simple rules [11]. Equipped with this greater understanding of emergence in population dynamics, we can better understand that practical solutions, like the Magic Roundabout, work because of underlying physical phenomena [12]. Thus, this formalization of collective motion and emergence has revolutionized the way we model biological populations, like the cell microscopy assays discussed in this thesis.

1.2.2 Agent-based Models

A fundamental objective of applied mathematics is the description of observable phenomena in terms of mathematical relationships. The process of determining these relationships is known as *mathematical modeling*, a term which encompasses a variety of different approaches. Many of these approaches are well-delineated through categorical descriptors: static versus dynamic, deterministic versus stochastic, etc. Yet the convention of viewing a dynamical system as an indivisible object, and modeling the only the summary behaviors of that system, is often taken for granted. An alternative approach is to treat the components of a system as the descriptive units.

Since we are interested in modeling collective motion systems, we want to include the aforementioned individual-level rules that result in emergent behavior. By modeling the units of a system as *agents*—autonomous actors which follow rules and make decisions—we allow these units to move about in a way that is both free and constrained by the rules we wish to impose. This is precisely the approach employed by *agent-based models* (ABMs). The model regards the behavior of the agents rather than the behavior of the system as a whole. How agents act independently and react both to other agents and to their environment influences the emergent behaviors of the system. This framework of agent-based modeling is not only natural, but also has certain advantages over typical *continuum* models, which instead model systems using continuous measures of population properties [13]. The primary advantage of agent-based models is that they provide an obvious way to model a multi-agent system in which more is known about the agents than about the summary behaviors of the system. This is the case in many models involving human agents; the action of an individual may be easier to model than the population's average action. Agent-based models are able to describe the complex population dynamics that arise from micro-scale interactions in a way that other mathematical models are not.

This benefit has proven advantageous when applied to cell movement and migration models. Driven by increasingly efficient and accessible computational tools for model simulation, the past decade has seen a surge of ABMs that more accurately describe how individual cells interact with each other and with the environment. Recent cancer cell migration models include terms that describe how cells respond to oxygen, drugs, growth factors, blood vessels (via angiogenesis) and other types of cells such as stem cells, and terms that describe a cell's propensity to adhere to other cells, proliferate via growth and division and move into unoccupied space [14]. Models describing cell migration into voids are now able to account for myriad factors such as cell adhesion, pushing, diffusion, and the approximate size of individual cells [15]. Furthermore, most ABMs can either be combined with relevant continuum models [16] or generalized to population-level models

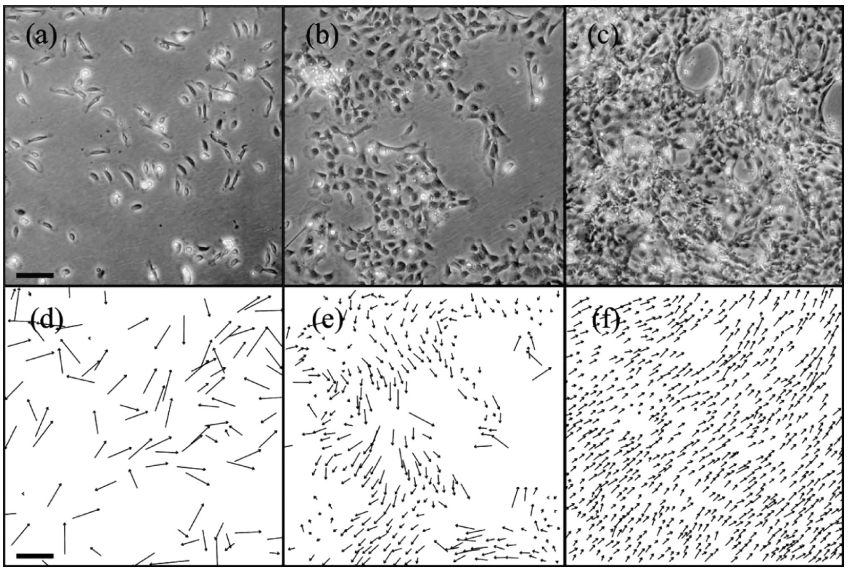

Figure 1.2: A figure from Vicsek and Zafeiris showing a collective motion model applied to cell migration data at varying densities [6].

using a method known as *coarse-graining* [17]. Due to these developments, ABMs are flexible for modeling the collective motion of cell populations, and they allow models to be devised from biological principles, yet implemented in a number of ways according to best fit.

The advantages of agent-based models are clear, mostly because they are exceedingly extensible and easy to conceive. A natural challenge arising from ABMs is that their output most always consists of multiple properties per agent, which are likely on the order of hundreds or thousands of outputs per observation. This contrasts sharply with continuum models, where even the most complicated models usually have only a few output quantities. There are a number of ways to summarize or otherwise alter the output of ABMs, but we most often present this resulting over-abundance of data in its original form as a *point cloud* of individual agents.

1.2.3 Point Cloud Data

An important first step in exploratory data analysis is determining the type of data with which one is working. The primary starting point is the distinction between categorical and quantitative data, which has been common parlance since at least the time of John Tukey [18]. Even within these classes of data, there are multiple ways to represent a dataset, and therefore multiple ways to perceive the same data. In fact, the manner in which a dataset is represented mathematically has an effect on how it may be analyzed.

For example, a dataset consisting of vectors can only have the inner product of those vectors computed if it is a subspace of a Hilbert space, though a collection of vectors can be thought of as a subspace of any vector space [19].

In particular, we think of each column or series of a multivariate numerical dataset as a dimension in multi-dimensional space. This is the intuition behind the common data science terminology of referring to individual data attributes as *dimensions*. To set observations as points in space is to represent a dataset as a *point cloud*. This is perfectly natural for data that are inherently spatial; landmarks on a map are an example of a point cloud in longitude/latitude "space". However, we may extend this idea to numerical data that are not inherently spatial. For example, one could plot the age, height, and weight of a person in 3-dimensional space. Doing this for multiple individuals would result in a "cloud" of points in that 3D space, and a way of representing this non-spatial numerical dataset in an innately spatial manner.

We formalize the notion of a point cloud by considering all numerical datasets as *finite metric spaces*. That is, we conceive of a dataset consisting of n instances of data with k properties as a collection of n points in k-dimensional space. Thus, the dataset is a multiset $\{p_1, p_2, ..., p_n\}$ where $p_i \in \mathbb{R}^k \ \forall i = 1, 2, ..., n$. Note that this is automatically a finite set given a finite number of observations. We then need only a *metric* which codifies the notion of distance between these points to make the set a finite metric space. As the usage of the set of real numbers suggests, we most often use the Euclidean metric in data analysis. In summary, we most often define point clouds as multisets of points $\{p_i\}_{i=1}^{N},\ p_i \in \mathbb{R}^k$ with the metric $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{j=1}^{k}(x_i^2 - y_i^2)},\ \mathbf{x} = (x_1, ...x_k), \mathbf{y} = (y_1, ...y_k)$.

Because each agent in an agent-based system is treated as a discrete unit, agent observations are easily conceived of as points in a point cloud. In particular, we use the k numerical properties that the agents have in common as the dimensions in k-space, and let each agent be represented by a point in the multiset. In this way we say that agent-based models directly describe the relationships between locations of points in a cloud, making plain the connection between ABMs and point cloud data. By describing ABM output in terms of point clouds, we are able to make a direct connection between ABMs and the data they endeavor to describe. The question of how to summarize and manage these seemingly unwieldy data remains, particularly for those with more than 3 dimensions, since they cannot be realistically visualized. There are matrix theoretic approaches to point cloud datasets, but most of these rely on decomposition techniques that perform dimensionality reduction at some step in the process. This essentially produces a *projection* of the point cloud into some lower-dimensional space and invariably results in a loss of information. Our approach, which is based on studying the shape of the point cloud as

a whole, aims to address this shortcoming, yet still provides concise summaries of the topological information of a dataset. By developing descriptiors that leverage the unique properties of point cloud data, we introduce a new method for estimating agent-based model parameters.

Chapter 2

Automatic Detection of Cell Nucleus Trajectories

2.1 Introduction

Biological researchers are increasingly leveraging cell trajectory data to better understand the underlying mechanisms of cell migration. Research shows the importance of collecting individual-cell measurements, as population-level assessment can mask important subpopulations within cell biology experiments. One study found that, even within clonal cell populations, the difference in stem cell antigen levels between cells varies enough to result in distinct subpopulations of cells that have entirely different transcriptomes, with certain proteins differing by multiple orders of magnitude [20]. This dissimilitude among clonal cells extends to cellular migration data as well, seeing as the movement of A549 cancer cells can only be accurately described by models that take individual cell behaviors into account [21].

Furthermore, the amount of variation in factors like chromosomal expression may be an important and useful quality of the cell population itself [22]. The notion of measuring this *cellular heterogeneity* and using it as a property of an experimental population is growing in popularity as statistics and biological sciences increasingly coincide [23]. These ideas, expanded in recent decades, highlight the need for collecting accurate, quantitative descriptions of individual cells from microscopy data. The tracking of each cell's movement over time is one such description. This process is known as *single-cell tracking*, even when those individual cells are among a population of hundreds or even thousands. In particular, by observing cellular motion *in vitro*, researchers hope to understand how to affect cell migration patterns, manage cellular morphology, and otherwise alter these cells

in vivo in order to develop and uncover novel treatments for wound healing and cancers, for which cell migration plays a central role.

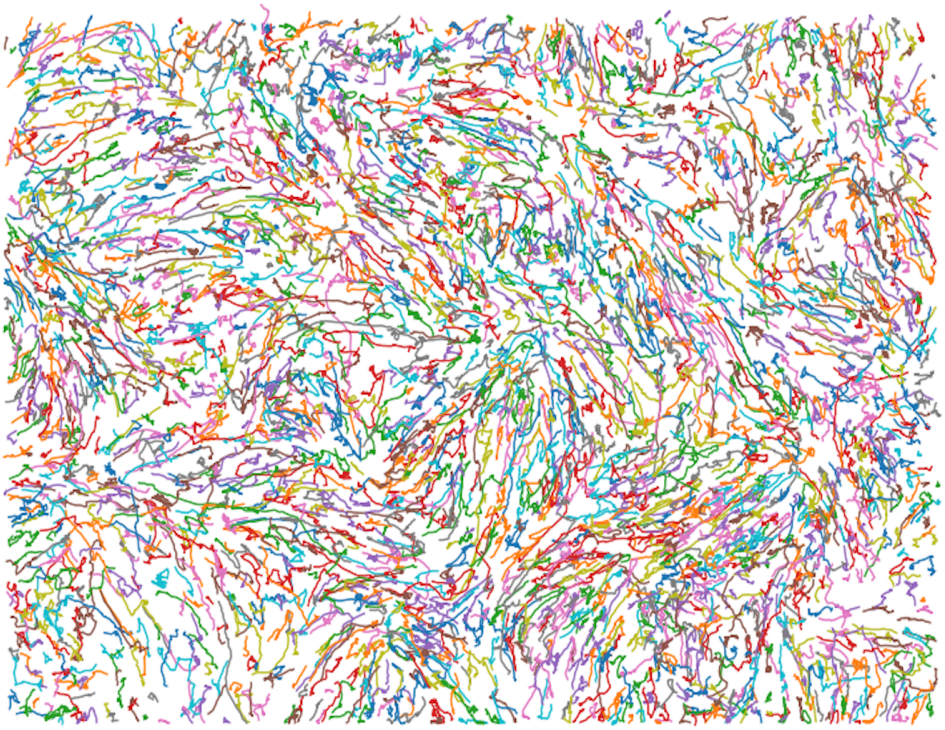

Figure 2.1: A visualization of tracked cell nucleus trajectories.

To this end, recent large-scale efforts have incentivized the application of emerging computer vision techniques to the task of automatically producing cell trajectories from raw cell microscopy imagery. Academically-supported projects ask computer vision researchers to develop novel data processing pipelines that can efficiently and accurately yield individual trajectories from diverse time-lapse cell microscopy datasets. The most successful and publicized of these efforts, the *Cell Tracking Challenge*, has produced over 20 open-source, peer-reviewed algorithms for segmenting and tracking cells in microscopy videos [24]. Recent years have also seen an increase in closed-source industrial cell tracking software, even implemented directly into lab equipment [25][26].

These approaches have continued to see widespread success, largely because most of these algorithms are based on *deep learning* [27][28], which is the current state-of-the-art for nearly all such computer vision tasks. In particular, the combination of convolutional neural networks (CNNs)—which are composed of many layers of nonlinear functions—and continued investigation into the extensibility of deep learning has revolutionized our

capacity to perform image classification, semantic segmentation, and object detection, which is the task that is most relevant to cell tracking. These complex CNN-based algorithms are the foundation of popular deep object instance detection algorithms like Faster R-CNN [29] and You Only Look Once (YOLO) [30]. Training these deep neural networks for the task of cell tracking, however, requires both an abundance of fully-annotated cell microscopy data and significant computational resources. Between data collection, cleaning, and annotating, it may take thousands of man-hours to build a dataset that is large enough such that one can get sufficient cell trajectories using these approaches. They also often require specialized hardware and computational tools such as high-performance general-purpose GPUs [27]. Training a neural network of this scale on standard laboratory computers can take weeks, and some algorithms will not even run on less sophisticated equipment. Accomplishing this process of data collection, annotation, and training for a new and novel dataset is often completely infeasible. Therefore a simpler approach from both a computer vision and experimental data collection standpoint is highly advantageous for practical cell microscopy research.

One recent focus of computer vision research toward this goal of applying deep learning algorithms to unseen datasets is the development of methods for *one-shot* or *few-shot* machine learning. The idea is to develop algorithms that learn from an exceptionally small number of training samples, as this allows for *data-lean* machine leaning strategies. The technique of applying deep CNNs to one-shot learning tasks is relatively new, and has been met with several challenges. One-shot computer vision algorithms are primarily based on sensing the differences between an anchor image and a new input image, an application that has seen much success in binary image classification such as facial recognition [31]. However the task of object instance detection—the first step in cell tracking—is much more complicated from a one-shot perspective. While new research has shown significant advances in this task, based on the two-step approach of the R-CNN methods and utilizing cutting-edge techniques like attention and excitation [32], its application to cell tracking remains to be explored. The primary concern with one- or few-shot approaches for cell microscopy image analysis is that these images often contain many samples. While most computer vision techniques are designed for image datasets with a relatively low, fixed number of objects per image, a single cell microscopy image may contain thousands of cells. Few-shot techniques would not only waste many of the data available, but are also much less likely to capture the variation between cells of the same class. We aim to develop an approach to cell tracking that does not rely on these few-shot approaches, but instead leverages the natural abundance of data present within a single image to strike a balance between data-lean requirements and full utilization of the available information.

Our approach combines these deep computer vision techniques with an image tiling approach, and hence requires only a single weakly-annotated training image. Our aim is to show that a simple deep convolutional neural network architecture applied to the task of cell tracking can provide significant improvements over classical computer vision algorithms without exceptional annotation requirements or computational burdens. We use a standard method for tracking objects through multiple frames of a video by first finding the locations of the objects in each frame, and then linking those locations together. A variety of approaches exist for both locating and linking, so this research focuses on the first step, and we rely on the published and proven Crocker-Grier method [33] for linking locations into trajectories.

2.1.1 Motivation

Our research is motivated by a novel scratch assay video dataset from the lab of Dr. Jason Haugh. These data show the migration and interaction of NIH 3T3 fibroblasts [34] over the course of several hours. Fibroblast migration is a central part of human wound healing, but its mechanisms of action are still rather poorly understood, primarily due to the wide variety of factors that contribute to fibroblast *taxis*, in contrast with well-investigated amoeboid cells [35]. While amoeboid cells generally move quickly and directly, fibroblasts and other mesenchymal cells move slowly and with weak polarization [35], and are more likely to experience inhibitory adhesion and contact [36]. In particular, various hypotheses exist as to the proportion that the different causes of locomotion—chemotaxis, haptotaxis, durotaxis—contribute to the directed migration of fibroblasts in different contexts [35]. Yet a better understanding of the cellular dynamics of wound healing could lead to advances in treatment that may save billions of dollars [37]. Furthermore, a limited comprehension of fibroblast activation has led to a lack of contemporary treatment options for fibrosis [38]. The research significance of accurate, analyzable fibroblast migration data is clear, as it could lead to breakthroughs in cell motion and interaction models and thus novel approaches to actual clinical treatment.

The videos provided by the Haugh lab enable a data-driven approach to fibroblast migration research. By quantifying observed dynamics, we can model local and global behaviors, and thus potentially uncover novel causes for migration and cell-to-cell interactions. If we are able to discern how observable behaviors correspond with specific migration cues, further research may be able to address the open question of which taxes cause fibroblast migration in different contexts. However, in order to begin this investigative process, we need reliable measurements of cell motion. Since manually

tracing these cell nuclei throughout this video dataset would be prohibitively expensive, we utilize a computer vision pipeline to process raw cell microscopy image data into individual-cell trajectories.

Our principal focus is to simplify this cell tracking process by considering only the location of each cell. Many "full-featured" tracking software packages find cells within an image by masking out the entire cell or cell nucleus, depending on the type of imagery. This allows for more in-depth study of cell microscopy including the analysis of cell morphology, and may improve an algorithm's ability to track the same cell over multiple frames, but requires orders of magnitude more training data—a cost that may be completely out of the question for academic research. For the sake of inexpensive, collective-motion-focused analysis, we only need point locations for each cell in each frame. Thus our goal is to recover only the *centroids*, or *centers of mass* (COMs), of the cell nuclei found in our dataset, rather than outlines or masks of entire cells. While the centroid of a fibroblast nuclei is not a perfect representation of the COM of the entire cell, mostly due to the complex morphology of fibroblasts, this estimate is a suitable surrogate for the location of the cell and has been successfully used in other research [39].

2.1.2 Cell Microscopy Dataset

We focus our efforts on a single set of fibroblast scratch assay videos provided by the Haugh Lab. This set of time-lapse images was collected using widefield epifluorescence microscopy to image NIH 3T3 fibroblast cells labeled with a far-red DNA-binding fluorescent marker which allows us to see the cell nuclei [40]. This particular dataset was obtained from a single dish containing two different densities of cells, which allows the study of how this density impacts cell migration. This level of density is measured via the *confluency*—a common metric of cell density for assay experiments that describes the percentage of the vessel or dish that is covered by cellular material [41]. We refer to the images that are 100% confluent as the "low-density" images, and the images which are densely-packed or *over-confluent* as "high-density". Note that "low-density" is somewhat of a misnomer, as microscopy experiments are routinely performed on cell monolayers that are less than 100% confluent. However, because scratch assays such as these almost universally use confluent monolayers [42], our use of the term "low-density" refers to this standard confluent density being lower than the over-confluent high-density portions of the vessel.

Within an hour of the images being collected, the cells are treated with platelet-derived growth factor subunit B (PDGF-BB). Thus the primary purpose of this data is to examine how fibroblast cell populations at different densities migrate in the presence of PDGF. A

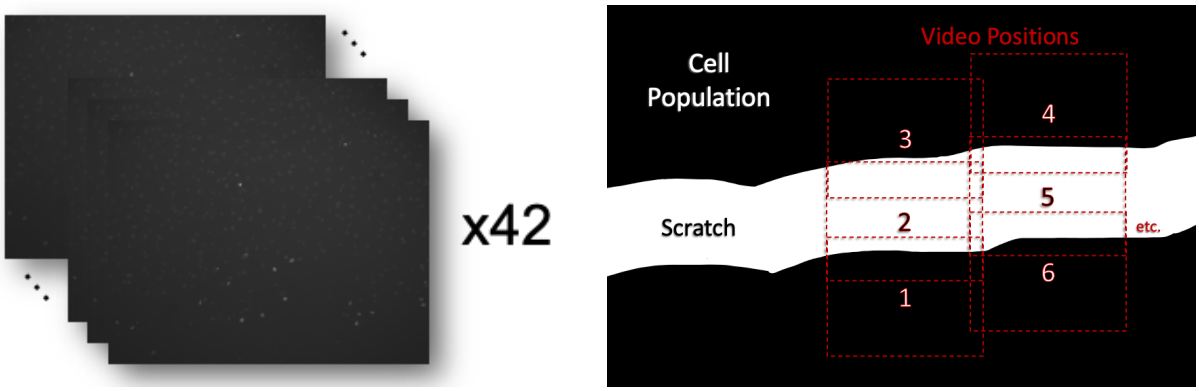

Figure 2.2: Diagrams illustrating the organization of the scratch assay image data. Each video is a sequence of 128 monochrome time-lapse frames and there are 42 videos in total (left). The videos are positioned over the scratch with partial overlap (right).

single scratch is made in the cell dish to imitate a wound, and images are collected at 42 distinct locations on the cell dish which we refer to as video *positions*. 18 positions show portions of the scratch where the cell density is high, focused at the top of the scratch, at the bottom of the scratch, or centered over the scratch such that there are cells visible both above and below (see Figure 2.2, right). Another 18 positions contain low-density cells with the same top, bottom, and centered scratch focuses. Finally there are 6 non-scratch positions, 3 with low cell density and 3 with high density, which we treat as a control group. The images in each of these four positional subsets—high-density scratch, low-density scratch, high-density control, low-density control—overlap on at least one side with another position from the same subset, making these data suitable for mosaicking into larger videos.

128 images are collected at each of these positions 10 minutes apart, resulting in the 42 time-lapse videos lasting approximately 22 hours. Thus the experiment results in 5376 individual images, which are acquired as single-channel, 16-bit TIFF images with a resolution of 1344 pixels by 1024 pixels (Figure 2.2, left). The spatial resolution is 1.157 pixels per micrometer (μm). Millimeters (mm) and hours (hr) provide a suitable combination of computational tractability and precision, so we use these as the units in our data analysis efforts. Any measurements collected at the pixel or frame level are hence converted to millimeters and hours before any consequent investigation.

2.1.3 Methodology Overview

The foundation of our automatic cell tracking pipeline is Trackpy [43], a Python package for automatically tracking particles through image sequences that uses an implementation of

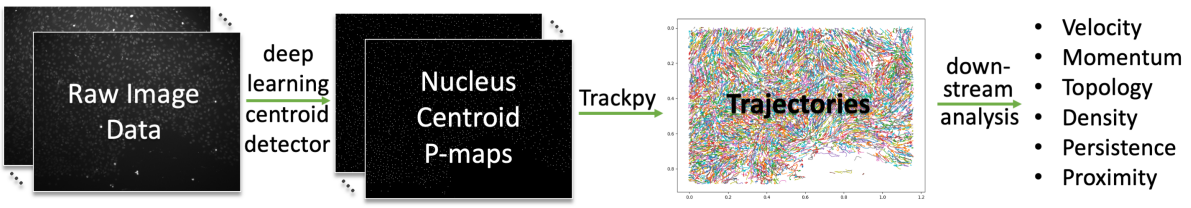

Figure 2.3: Cell tracking pipeline overview.

the Crocker-Grier algorithm [33]. Trackpy operates on image data using three algorithmic stages: feature location, coordinate refinement for the sake of sub-pixel accuracy, and linking of particles into trajectories [44]. Preliminary attempts to use Trackpy on our dataset were lacking, primarily due to missing cells or false detection in individual frames (see Figure 2.4), leading to short and inaccurate trajectories. To overcome this issue we focus not on developing a tracking algorithm from scratch, but rather on developing a method for **nucleus centroid detection**. Once we have successfully located cell centroids, we pass this information to Trackpy with the aim of improving particle detection. In this sense, our contribution is a deep-learning-based preprocessing method for finding cell nuclei in raw imagery. We will herein present the details of our centroid detector before discussing the specifics of the Trackpy algorithm. The cell tracking pipeline does, of course, include both the collection of cell microscopy data as discussed above, and the analysis of the resulting trajectories. This entire end-to-end process is outlined in Figure 2.3.

2.2 Centroid Detection

Motivated by the success of deep convolutional neural networks (CNNs) at nearly all computer vision tasks over the past decade, we aim to develop a centroid detector that is at once informed by the nature of our cell microscopy dataset through training data, and yet simple enough to be both computationally inexpensive and robust to differences between the frames in the dataset. Furthermore, we acknowledge the need for only point locations of the cell nuclei in each image, an endeavor that should be much simpler than the common computer vision task of predicting bounding boxes or outlines around each object in an image.

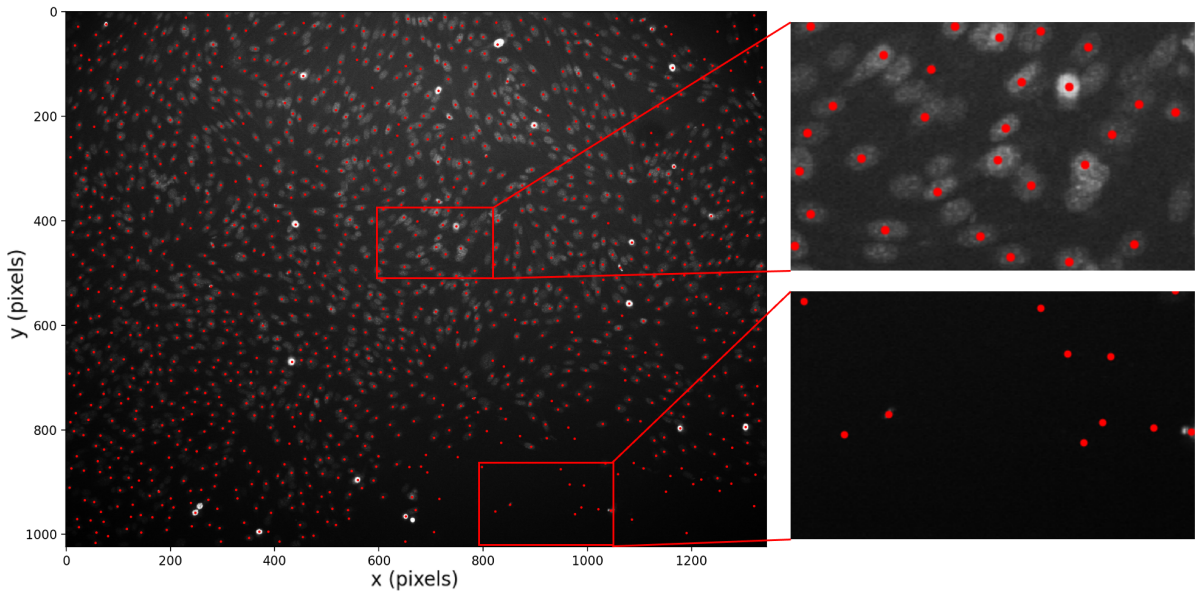

Figure 2.4: Cell nucleus point locations (red) detected using Trackpy with no image preprocessing. Plotted on a single cell microscopy frame, contrast corrected for legibility. Crops show missed nuclei (top right) and false detections (bottom right).

2.2.1 U-Net

Because of our specific requirements of point location detection and data efficiency, we eschew the aforementioned popular deep computer vision processes based on YOLO [30] and R-CNN [29]. While these approaches have proven to be remarkably effective at detecting a variety of objects at a variety of scales given sufficient training data, our dataset consists of relatively uniform objects at a single scale, and no pre-labeled data to be used for training. We instead develop a new method based on *U-Net*, a deep neural architecture developed for biological image segmentation tasks similar to the one we face.

The U-Net architecture was introduced in 2015 by Olaf Ronneberger and colleagues in an attempt to improve deep neural net image segmentation performance while leveraging a smaller amount of annotated training data [45]. This fully-convolutional architecture works by concatenating feature maps from the down-sampling stage of the network to the corresponding up-sampling feature map (see Figure 2.5). The result is a network that has all of the feature-finding capabilities of a deep architecture yet does not sacrifice the high-resolution information provided by the input image. As a consequence of both the architecture and the abundant application of training image augmentation the network is highly robust to noise, making it ideal for data-limited computer vision applications.

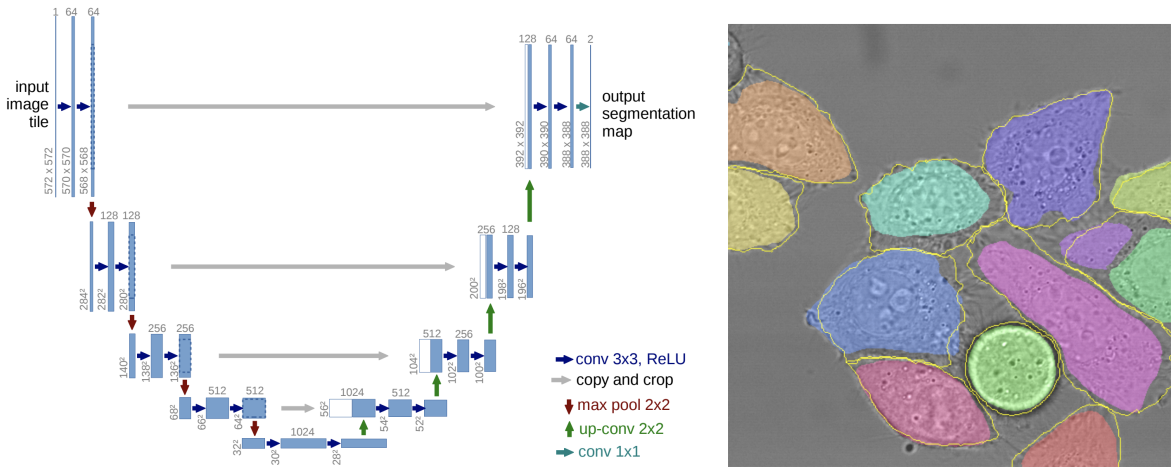

Figure 2.5: The network architecture (left) and example segmentation result (right) from the original U-Net paper [45].

U-Net Loss Function

The original U-Net is tasked with providing *semantic segmentations* of biological images, where each pixel in an image is labeled according to its class. Using classical machine learning notation, we say that the neural network is attempting to learn the function $f : X \to Y$ where $\mathbf{X} \in X = \mathbb{R}^{h \times w}$ is a single-channel input image and $\mathbf{Y} \in Y = \mathbb{R}^{a \times b \times k}$ is a *segmentation map* identifying each pixel of the a-by-b sub-image of the input as a member of one of k classes. We train the network such that the *loss function* $L : Y \times Y \to \mathbb{R}$ is minimized for all elements of the *training set* $T = \tilde{X} \times \tilde{Y}$ where $\tilde{X} \subset X,\ \tilde{Y} \subset Y$:

$$\begin{gathered} \min L(\mathbf{Y}, \hat{\mathbf{Y}}), \\ (\mathbf{X}, \mathbf{Y}) \in T,\ \hat{\mathbf{Y}} = f(\mathbf{X}). \end{gathered} \tag{2.1}$$

From here we take unseen images $\mathbf{X} \in X - \tilde{X}$ as input to get predicted segmentation maps $\hat{\mathbf{Y}} = f(\mathbf{X})$, hoping that the function f has accurately learned how to map images to segmentations.

From a traditional statistical learning perspective, classification is a discrete process which consists of placing inputs into a finite number of categories. The U-Net task of *pixel classification* is thus non-trivial for neural networks, whose training procedure relies on every layer being differentiable. This disallows a direct application of neural networks to classification tasks, as a discrete output layer would be non-differentiable. To overcome this, neural networks leverage a discrete probability distribution output where the ith element of the sample space corresponds to the input being a member of the ith class. This

extends to any number of categories, with k-class classification using a sample space with k values. These probabilities are continuous, and thus can be differentiated with careful construction of the function that maps the learned features to the output probabilities. The probability distribution output is created using the *softmax* function, a differentiable approximation of argmax that can be understood as the k-dimensional extension of the sigmoid function:

$$g_i(\vec{y}) = \frac{e^{\vec{y}_i}}{\sum_{j=1}^{k} e^{\vec{y}_j}} \tag{2.2}$$

where $\vec{y} \in \mathbb{R}^k$ [46]. By this definition, $g(\vec{y}) \in \mathbb{R}^k$ as well, so the k-dimensional vector $g(\vec{y})$ is precisely our desired probability distribution. By applying the softmax *pixel-wise* to the entire penultimate feature map, we obtain a vector of the learned probabilities of any pixel being in each class ($\hat{y}$) and thus recover our desired segmentation map:

$$\hat{\mathbf{Y}}_{r,c,*} = \hat{y}_{r,c} = g\left(\bar{f}(\mathbf{X})_{r,c,*}\right) \tag{2.3}$$

where $\bar{f}$ returns the last non-output feature map of the neural network—the map of values given before the softmax is applied.

At training time we need a measure of similarity between these k-dimensional probability distribution vectors which are predicted for each pixel ($\hat{y} \in \hat{\mathbf{Y}}$), and the ground truth *one-hot labels* for each pixel, which are simply the standard basis vectors corresponding to the ith category:

$$\begin{gathered}(\mathbf{X}, \mathbf{Y}) \in T, \\ \mathbf{X}_{r,c} \in \text{category } i \Rightarrow \mathbf{Y}_{r,c,*} = \mathbf{e}_i.\end{gathered} \tag{2.4}$$

This measure of similarity will act as the loss for each pixel, so in particular we need a real-valued function ℓ such that $\ell(\mathbf{e}_i, \hat{y}) \in \mathbb{R}$ where $\mathbf{e}_i,\ \hat{y} \in \mathbb{R}^k$ when we have k classes. The function used almost universally in classification tasks is the *cross-entropy loss* [46]:

$$\ell(\vec{y}, \hat{y}) = \sum_{i=1}^{k} -\vec{y}_i \log \hat{y}_i \tag{2.5}$$

where $\vec{y}$ is a ground-truth label standard basis vector as above. While the name comes from information theory, this loss function is the extension of the log-likelihood used in univariate logistic regression [47], and in fact results in maximum likelihood estimates for the parameters of the neural network under certain assumptions [46]. Altogether, the combination of softmax and cross-entropy results in neural networks essentially performing generalized multi-class logistic regression on each of the pixels in the output map for the

task of classifying said pixels.

In practice, a machine learning algorithm needs a single value to minimize ($L(\mathbf{Y}, \hat{\mathbf{Y}}) \in \mathbb{R}$), but by applying cross-entropy pixel-wise we currently have a real-valued map of entropies $M \in \mathbb{R}^{a \times b}$. The standard approach is to simply sum all the entropy values in the entire map to get a single value to minimize [48]:

$$L(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_{r=1}^{a} \sum_{c=1}^{b} \ell \left(\mathbf{Y}_{r,c,*}, \hat{\mathbf{Y}}_{r,c,*} \right). \tag{2.6}$$

However this results in treating the accuracy of each pixel in the input image with equal importance. As this is often not the case, the original U-Net extends cross-entropy slightly to perform *weighted* pixel classification. In particular they assign a higher importance to pixels that act as the boundary between cells in an attempt to improve the network's ability to distinguish between instances of the same class of object (see Figure 2.5). This is done with a simple pixel-wise weight map $\mathbf{W} \in \mathbb{R}^{a \times b}$ with a higher value causing that pixel in the sub-image to be more strongly-weighed in the loss function [45]:

$$L(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_{r=1}^{a} \sum_{c=1}^{b} \mathbf{W}_{r,c} \, \ell \left(\mathbf{Y}_{r,c,*}, \hat{\mathbf{Y}}_{r,c,*} \right). \tag{2.7}$$

U-Net Optimization

The actual training of the algorithm consists of minimizing this loss via first-order gradient descent performed on all the network parameters simultaneously—a process known as *back-propagation* [46]. The actual gradient descent update iteration for a parameter w is done by taking a fixed step of length γ—called the *learning rate*—in the direction of the derivative of the loss with respect to the parameter [46]:

$$w_{j+1} = w_j - \gamma \frac{\partial L}{\partial w}. \tag{2.8}$$

This is most commonly done in batches of inputs by adding the gradient value for each input before taking a step, and in randomly-shuffled orders of the inputs. This variant, which is called *stochastic gradient descent* (SGD), reduces the chance of falling into a sub-optimal local minimum and often speeds up training. As a simple local optimization technique that can handle large datasets and an often-massive amount of model parameters, SGD performs well enough that it—as a base minimization method— has become the standard for deep learning [49].

One relevant SGD variant is the addition of *momentum*, which influences the direction

Figure 2.6: Graphics showing the convergence of gradient descent without momentum (left) and with momentum (right) [50].

of the gradient update step through a process analogous to physical momentum [51]. Momentum works by introducing a velocity term v that, by keeping an exponentially-decaying average of past gradients, keeps the update step moving in the same direction. In practice, this is done by first updating the velocity in terms of both the momentum—parameterized by α—and the new gradients using a normal learning rate γ, and then by updating the parameter using the velocity. Thus, rather than the standard gradient descent step in (2.8), we may use a momentum update:

$$\begin{aligned} v_{j+1} &= \alpha v_j - \gamma \frac{\partial L}{\partial w}, \\ w_{j+1} &= w_j + v_{j+1}. \end{aligned} \tag{2.9}$$

The details of how this corresponds to physical momentum can be found in Goodfellow's seminal work on deep learning theory [51]. In practice, momentum simply keeps the update steps moving in the correct direction, helping to counteract the effects of noisy or small gradients (see Figure 2.6).

2.2.2 Novel COM Detection Algorithm

In cases where the members of a class are rather distinct in terms of visible features or where there are many classes per image, it may be possible for U-Net to distinguish each object from others nearby due to the use of boundary pixel weighting in the cross-entropy loss. However, because we have only one class of object—cell nucleus—and many visually-similar nuclei in very close proximity, we cannot rely on semantic segmentation to provide sufficient distinction between objects. We need to be able to detect each individual *instance* of a cell nucleus in our images, but would like to retain the robustness and efficiency of the U-Net architecture.

Probability Map Nuclei Detection

We solve the problem of converting this semantic segmentation algorithm to an instance detection algorithm by having the network output instance locations as points, rather than as class segmentation maps. To this end we target narrow Gaussian distributions at the centroid of each cell nucleus, allowing these to represent the point location of each cell. The result is a network which outputs maps that contain these Gaussian peaks, often called *p-maps*. This approach has been shown to be particularly effective at detecting cell nuclei in microscopy data [52]. Using a sufficiently small value for the uniform variance (σ^2) of the Gaussians gives peaks that are narrower than the minimum cell nucleus size, providing suitable separation between nuclei for instance detection. For our data we use $\sigma^2 = 2$ px^2, which provides sufficient separation for adjacent cells (see bottom left of Figure 2.7). Specifically, we create our targets for training by obtaining the value of the PDF of the normal random variable centered at the target COM at each pixel in the map for each cell, and taking the max pixel value over all such single-cell maps. Thus, given the PDF of a bivariate normal random variable with mean $\mathbf{c}$ and variance σ^2 in both directions

$$f(\mathbf{x};\mathbf{c}) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{c})^{\mathrm{T}}(\sigma^2\mathbf{I})^{-1}(\mathbf{x}-\mathbf{c})\right]}{2\pi\sigma}, \tag{2.10}$$

we obtain a target p-map $\mathbf{Y} \in \mathbb{R}^{a\times b}$ thusly:

$$\begin{aligned} (\mathbf{P}_i)_{r,c} &= f([c,r]^T;\mathbf{c}_i), \\ M &= \max_{\forall i,\forall r,\forall c}\left\{(\mathbf{P}_i)_{r,c}\right\}, \\ \mathbf{Y}_{r,c} &= \frac{1}{M}\max\left\{(\mathbf{P}_1)_{r,c},...,(\mathbf{P}_N)_{r,c}\right\} \end{aligned} \tag{2.11}$$

where the $\mathbf{c}_i \in \mathbb{R}^2,\ i = 1,...,N$ are the pixel coordinate locations of the ith cell centroid included in the p-map for all of the N cells.

P-maps have several advantages over the semantic segmentation task for our COM detection purposes. Firstly, the creation of target p-maps is less demanding from a labeling standpoint, since target maps can be created from simple point annotations. We may therefore refrain from creating segmentation masks or bounding boxes for our training data. Creating a p-map using Gaussian distributions also affords a richer description of location, as these distributions create a theoretically infinite range of non-zero values. This is, of course, not practically the case, as floating-point arithmetic has a floor of precision according to how many bits are used. However, the Gaussians still provide a measure of proximity that aids the training of the network [52]. Finally, the p-map

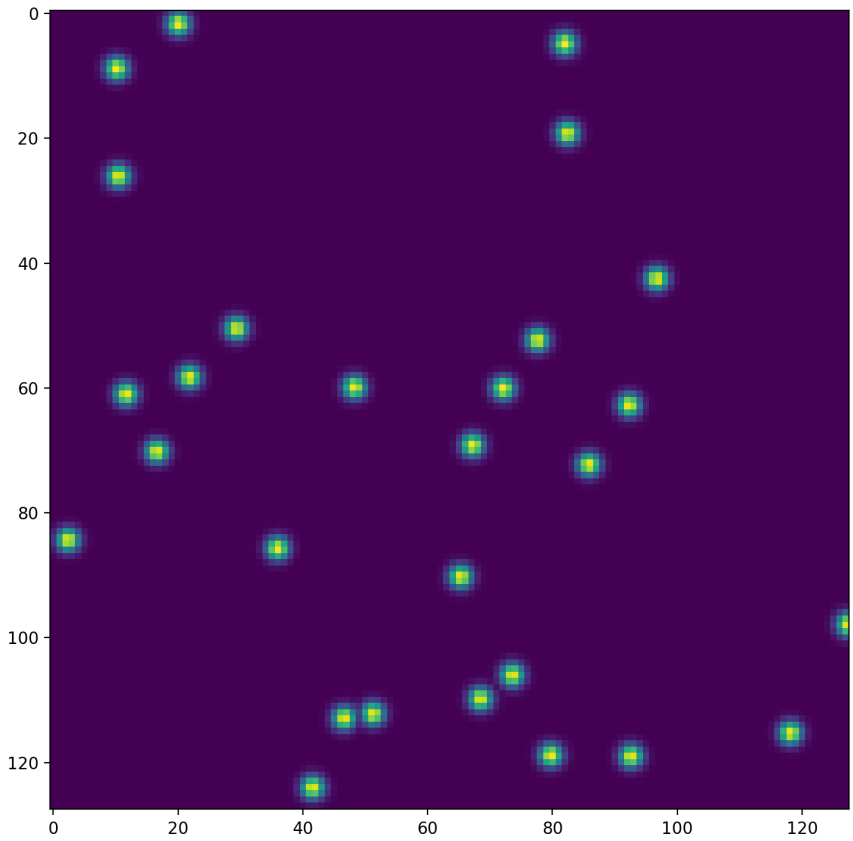

Figure 2.7: A sample target p-map. Note the irregular peak shape caused by non-integer centroid locations, which allows for sub-pixel location accuracy.

approach allows for sub-pixel accuracy, as the cell centroids ($\mathbf{c}_i$) need not be integer values (see Figure 2.7). While a pixel-based approach is usually sufficient for segmentations, requiring a cell's point location to be fixed to a grid of the same resolution as the output map is a serious detriment to downstream analysis. Our primary concerns are a loss of accuracy in the calculation of cell properties—such as velocity and nearest-neighbor proximity—that require precise positions, and the consequences on modeling efforts, since integer positions where only one cell can occupy each location necessitate a lattice-based approach to agent-based modeling.

Centroid Detector Loss Function

This approach of single-channel p-map outputs also removes us from the classification neural network perspective, and allows us to consider the problem as one of pixel-wise *regression*. The optimization task thus becomes the straightforward minimization of the distance between continuous pixel values, rather than the optimization of a measure

of difference between continuous approximations of discrete categories. Because the difference between pixels creates a matrix of residuals which must be aggregated, this scalar distance can be computed in a number of ways. We use a standard approach of taking the mean squared error (MSE) between the prediction and target and using it as the loss function [46]. Thus, for our target p-maps $\mathbf{Y} \in \mathbb{R}^{a \times b}$ and predicted p-maps $\hat{\mathbf{Y}} = f(\mathbf{X}) \in \mathbb{R}^{a \times b}$, we have

$$L(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{ab} \sum_{r=1}^{a} \sum_{c=1}^{b} \left(\mathbf{Y}_{r,c} - \hat{\mathbf{Y}}_{r,c} \right)^2. \tag{2.12}$$

Note that this loss function, unlike cross-entropy, does not require a specific probabilistic interpretation of the inputs, and thus does not depend on a particular activation function like a softmax. In fact, MSE loss works with any output that allows for values that are the same as the target. In our case, because we have normalized our targets to take values in $[0, 1]$ by dividing each element of the matrix by the maximum value, we also want to limit our output values to the same range. This is most commonly done using the sigmoid activation function [46], a basic logistic function which takes any real number as input and returns real numbers in $(0, 1)$:

$$g(x) = \frac{1}{1 + e^{-x}}. \tag{2.13}$$

While this function cannot theoretically attain the values 0 and 1, the limitations of floating point arithmetic allow our neural network to output either value with sufficiently large positive and negative values in the penultimate feature map. The sigmoid activation function also has the benefit of having the steepest gradient near inputs of zero which keeps the network from learning to predict values in between the background, which has an output near 0, and the peak values, which have outputs closer to 1. Thus, parallel to the presentation of the U-Net classification network, we apply the sigmoid pixel-wise to the output, letting

$$\hat{\mathbf{Y}}_{r,c} = g\left(\bar{f}(\mathbf{X})_{r,c} \right) \tag{2.14}$$

where $\bar{f}$ once again yields the second-to-last feature map of the network.

Image Data

We train our centroid detector using a standard "data-sufficient" deep learning approach disguised as one-shot learning. Whereas one-shot machine learning uses only one sample to teach an algorithm how to complete a task, we sample tiles from our single training

image many times to form a large dataset of tiles. Specifically, our network takes 128-pixel by 128-pixel tiles of a single-channel cell microscopy image as input, and yields p-maps revealing the location of each cell nucleus in the image tile. This approach has several advantages over the standard deep computer vision approach of taking entire images as input, the most significant being that it greatly increases the amount of available training data.

At each of the 5000 training epochs we first uniformly sample 2000 tiles from the training image, and then apply between 0 and 3 image augmentations to each tile. We allow three square symmetry augmentations: flips in either perpendicular direction and a single rotation to any of the rotational symmetries. In this way we allow for all possible square symmetry augmentations while minimizing the probability of recovering an unaugmented tile after augmentation has been applied; the combination of both perpendicular flips and a 180 degree rotation is the only way to obtain the original tile. We also include the option to apply affine shear or Gaussian blur to the image, which compensates for the jitter and lack of focus that occur as part of the data collection process.

We justify the use of a single training image by comparing the number of tiles needed for our full set of training iterations to the number of possible base tiles that may be created from the image with our augmentation configuration. Our cell microscopy images have a width of 1344 pixels and a height of 1024 pixels. Thus there are $(1344-128)(1024-128) \approx 1.09\times10^6$ possible tile locations. Each tile may have up to 3 out of 5 distinct augmentations applied, ignoring the degrees of freedom within each augmentation, so the total number of possible "base versions" of each tile is

$$\binom{5}{0}\binom{5}{1}\binom{5}{2}\binom{5}{3} = 1 \cdot 5 \cdot 10 \cdot 10 = 500.$$

In the end, we have a collection of more than 500 million base training tiles from which to sample. Because we train the network for 5000 epochs, we need a total of 10 million tiles. These 10 million tiles required pale in comparison to the distinct tiles available, in fact less than 2% of the base tiles will be used during this training configuration, suggesting we need not worry about exhausting our training data and overfitting to repeated samples. The result of this tiling approach is an abundance of data for training an adequately-deep neural network for the task of centroid detection. Yet from the operator's perspective, only one annotated image is needed to successfully process an entire image dataset, providing all of the benefits of a one-shot approach without any of the statistical drawbacks of training data limitations.

Network Architecture and Details

This method of training on fixed-size tiles taken from a larger image is similar to the approach of the original U-Net. Since our network accepts and outputs tiles that are far smaller than an entire cell microscopy image, we deterministically "slide" the network over the whole image at inference time, combining the resulting p-map tiles (similar to Figure 2.7) to make a p-map of the entire input image (Figure 2.8).

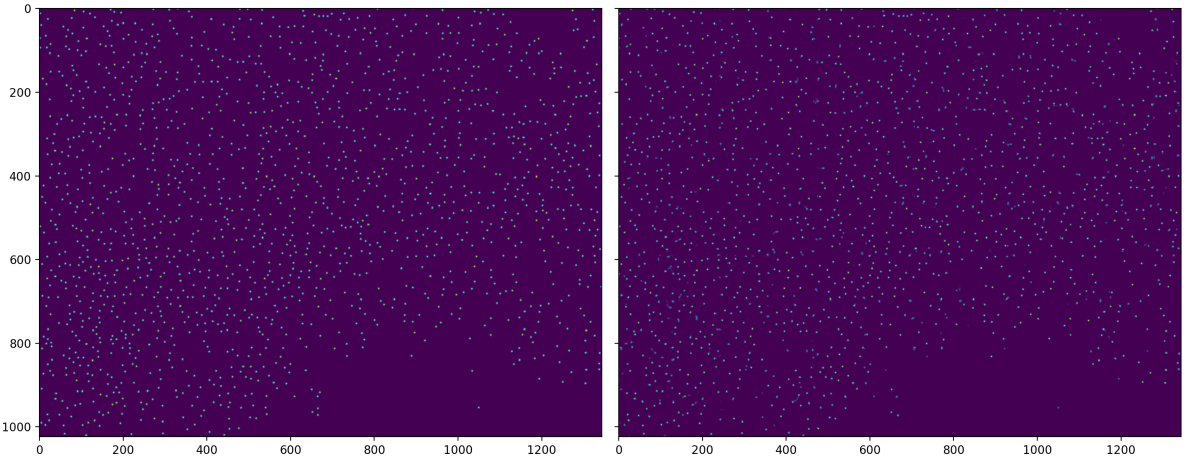

Figure 2.8: The ground-truth p-map (left) and predicted p-map (right) for an unseen test image.

The centroid detector network architecture is a variation of the signature U-Net design, with 4 stages of 2-dimensional downsampling convolutions and 4 stages of 2-dimensional upsampling *transpose convolutions* [53]. The final feature map of each downsampling stage is concatenated to the input feature maps of each upsample stage, which is the foundation of the original U-Net [45]. The result of these 4 stages is dimensional reduction by a factor of 2^4, resulting in a minimum feature map dimension of 8x8 pixels. We also increase the number of feature maps by a factor of 2 at each downsampling stage, from 32 initial feature maps after the input convolution to 512 feature maps at the highest level of abstraction. The full network architecture diagram is included in the appendices (Figure A.1).

Significant differences from the original U-Net include a lack of dropout layers, as their inclusion results in an increased number of missed detections in our experiments, and is cited as reducing performance and increasing training time [52], and the use of "same" padding at each convolution, which disposes of the need to crop at each concatenation.

The removal of dropout is standard for this type of homogeneous dataset, since dropout is a regularization technique and only serves to reduce overfitting. Because U-Nets are entirely convolutional, standard dropout does not help regularize the network in the way that it has been proven to for fully-connected layers [54]. Furthermore, by using a sufficiently-simple architecture and ample training data augmentations, we achieve enough generalization to perform well on unseen data without needing any type of dropout.

Lastly, we compensate for the lack of cropping in the architecture by allowing the Gaussian tails of centroids outside a tile to "bleed over" into the tile, resulting in a significant signal near the same-padded boundary. We accomplish this by picking an *enlargement percentage* ξ, which uses cell center locations outside of the tile to create the target p-map. No enlargement would result in only locations within the tile being included as Gaussian peak means, while a positive enlargement $\xi > 0$ evaluates the normal PDF for the exterior centroid locations within ξ percent of the boundary. To be precise, given an enlargement percentage $\xi \geq 0$ and the set of all cell centroids $\mathbf{C}$ in terms of the pixel coordinates of a given tile $\mathbf{x} \in [0, a] \times [0, b]$, select $\mathbf{c} \in \mathbf{C}$ in case

$$(-\xi b \leq \mathbf{c}_1 \leq b + \xi b) \wedge (-\xi a \leq \mathbf{c}_2 \leq a + \xi a). \tag{2.15}$$

The non-zero values in the target p-map caused by this enlargement teach the network to look for and detect centroids at tile boundaries by refusing to suppress weights at the edges of feature maps. To further ensure no performance is lost at tile edges which are located inside the image we also use a slight overlap of tiles at inference time. The exact amount of overlap does not appear to be particularly impactful, but may be tuned inexpensively since it does not impact training. We use overlap of 20% for our final experimental p-maps.

Finally, we train the network for 5000 epochs using stochastic gradient descent with the learning rate fixed at 0.05, and use momentum of 0.9, which aims to smooth oscillations in the weights during training and train to a better local optimum. To summarize, our centroid detector algorithm consists of a deep U-Net-type neural network taking cell microscopy image tiles as input and outputting tiles of p-maps showing the centers of the pictured cell nuclei, which is in turn wrapped in processing code that tiles a larger input input image and outputs the corresponding whole-image p-map, stitched together from output tiles. This step of the process is trained end-to-end from scratch in order to yield the best possible centroid locations for a given dataset. We find that this approach is both conceptually uncomplicated and visually reasonable: given an image, predict the most probable locations of object centers. We will also show that it is sufficiently robust

to the noise present in the dataset. Using these p-map outputs as input into Trackpy in place of the raw imagery, we expect to see a significant improvement in performance.

2.3 Tracking Cell Nuclei

Equipped with accurate cell nucleus locations for each frame of our video datasets in the form of p-maps, we now focus on the task of linking these locations into usable cell trajectories. While there are many approaches to particle linking, from nearest-neighbor approaches to multi-frame dynamic programming, we utilize the implementation of the Crocker-Grier nearest-neighbor algorithm provided in the Trackpy software package. While simple, the Crocker-Grier approach is both efficient and effective, and performs well on cell tracking tasks when compared to more complicated algorithms [55]. Furthermore, Crocker-Grier as packaged in Trackpy includes precise sub-pixel location functionality, which is ideal for converting our p-maps into coordinates for the sake of linking.

2.3.1 Crocker-Grier Algorithm

The particle tracking algorithm published by John C. Crocker and David G. Grier in 1996 divides the task of recovering particle trajectories from raw image data into 5 stages: image restoration, locating particles, refining location estimates, noise discrimination, and linking locations into trajectories [33]. The Trackpy implementation presents this as 3 simplified steps based on an IDL implementation [56] of the original: feature finding, refinement, and linking [44]. This algorithm is ideal for processing our p-maps into trajectories, as it is based on finding Gaussian-shaped spots in an image and using the properties of Gaussian functions to get ideal particle locations to link into trajectories.

The first step in the process is to preprocess the image in order to highlight true features and remove background noise. While this is far less necessary for our p-maps—which may in some ways be considered a noiseless, synthetic form of data—than it is for real imagery, we still find that it improves the end result. This is likely due to the artifacts produced by stitching the individual tiles together into an entire-image p-map, particularly apparent when cells are near the edge of a tile. The preprocessing consists of a standard band-pass filter, common in signal processing applications, implemented as a single convolution of the image with a combination Gaussian-boxcar kernel [33]. This operation suppresses sufficiently-dark background pixels while keeping the bright spots at the same intensity, effectively defining the edges of bright Gaussian-shaped objects (see Figure 2.9).

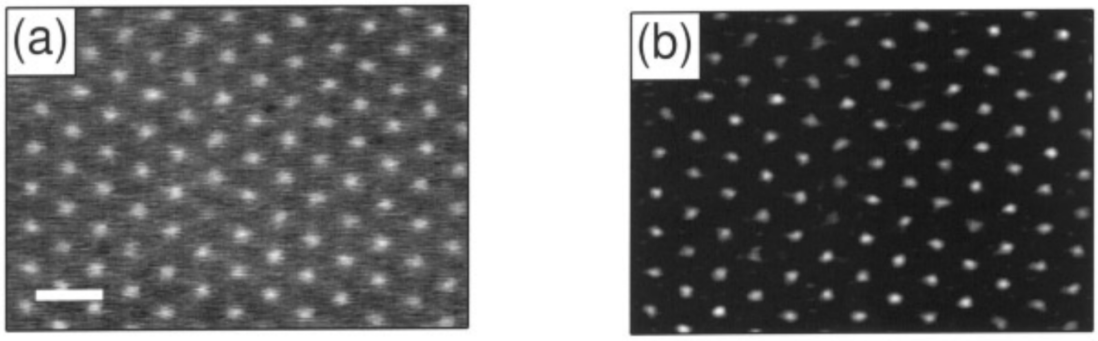

Figure 2.9: Figure from the Crocker-Grier paper showing the effects of the band-pass convolution [33].

The next step is the identification of the pixel coordinate of the center of each particle. Because the locations are modeled as Gaussian peaks, this is done by simply finding the brightest pixel using gray-scale dilation [33]. Since the goal, however, is sub-pixel accuracy, these are treated only as *candidate* particle locations. These candidates are refined by looking for the center of mass of the peak. The mass is approximated by adding the intensities of the pixels within a certain radius of the candidate pixel. The actual particle location is determined by the center of this mass approximation, so the candidate is moved closer to it, changing candidate pixels entirely if needed. This process is done iteratively in practice, up to either a maximum number of iterations or until the iteration step is sufficiently short [57].

The end of the location process consists of several filtering components. The primary filter is the removal of locations that are too close together based on expected particle diameter, which Trackpy requires as a user input into the algorithm. Other filtering operations are based on the noise discrimination moments computed alongside the particle mass in previous steps. Besides the particle mass, these moments include the signal and signal-to-noise ratio, and the eccentricity [58]. We do not utilize the optional mass- and signal-based filtering, and instead focus on tuning the required diameter parameter. Trackpy does, however, allow users to specify all the parameters of both the standard and optional filtering and processing components of the Crocker-Grier locating procedure.

Finally, armed with accurate, sub-pixel locations for each of the particles in the image, we can link them together into trajectories using nearest-neighbor association. The Crocker-Grier algorithm utilizes an efficient nearest-neighbor procedure based on forming connected subnetworks of particle locations for each particle using a sufficiently-small search radius [33]. This process essentially consists of linking each particle in a given frame to the particles in the next frame that are within a user-prescribed radius L. If

the radius is small enough, this splits the entire network of particles into disconnected subnetworks which can be examined independently, and even in parallel. If there is only one nearest neighbor connection between particles in consecutive frames, these are linked. Otherwise, the subnetwork is linked iteratively using the shortest connections and adding connections longer than L if there is a discrepancy between the number of current and future particles in a subnetwork that would lead to "orphaned" particles. If the choice of L is sufficiently short, this process approaches linearithmic time complexity, making it one of the quickest options for linking [33].

Crocker and Grier do discuss the limitations of using nearest-neighbor linking on interactive or locomotive particles, as the theoretical guarantees of a particle being its own nearest neighbor are founded on noninteracting particles undergoing Brownian motion [33]. In short, they recommend using a time scale fine enough that the average particle moves less than the typical distance between particles, and suggest using a search range L of less than half that typical distance whenever possible. Our cells do violate the first recommendation at times, particularly those that move quickly and with persistent direction. However, the Crocker-Grier paper also acknowledges that mislabeled particles rarely limit the practical investigation of trajectories, which we aim to show with further analysis.

The Trackpy implementation of Crocker-Grier linking includes several extensions to the nearest-neighbor approach not leveraged in the original paper. These include a memory function for particles that briefly disappear, various and adaptive strategies for forming subnetworks and searching them for neighbors, and a prediction framework which looks for successive particles near it expected location rather than its current location [59] The only one of these features we utilize is the memory, which helps to compensate for blur that differs between frames of the same video.

Altogether, the Crocker-Grier particle tracking algorithm provides a complete process for extracting cell trajectories from image data, and is ideal for p-maps since it is based on locating Gaussian peaks. While the algorithm consists of several mathematically-justified steps, it can be summarized as a two-stage process of detecting particle centers of mass and subsequently linking them together into trajectories by looking for nearest neighbors. The original paper provides many more details on the development of the algorithm, the error bounds based on time scale and magnification level, and applications to experimental spherical particle data [33]. The source code for Trackpy also details some of the extensions to the methodology, and details some of the implementation specifics [60]. We obtain trajectories for our cell microscopy videos by applying Trackpy's Crocker-Grier algorithm to the p-maps resulting from the previously detailed centroid detection process.

2.3.2 Cell Tracking Methodology

Because our investigation hinges primarily on the effect that p-map image processing has on the ability of an algorithm to accurately track cells in a microscopy video, we utilize the aforementioned Trackpy implementation of the Crocker-Grier algorithm [60] for all subsequent tracking steps. We utilize the *batch* and *functions*, which locate the cells in an entire sequence and link together locations into trajectories respectively, to acquire base trajectories. We then subsequently improve the trajectories by using *filter_stubs* to remove trajectories that are too short to be useful, hoping to catch any false detections, and by using the *compute_drift* and *subtract_drift* functions to remove the movement of the imaging device relative to the cell dish. In the end, Trackpy returns extensible Python pandas DataFrame objects [61], which provide easy ways to evaluate results, options for further analysis, and several methods for saving and exporting.

We provide sequences of p-maps—one for each time step in the experimental dataset—to the Trackpy batch locating function in the requisite 8-bit single-channel image format. We use a diameter of 11 pixels based on the average size of the cells in the images. The rest of the particle locating options, including pre-link filtering methods, are ignored in favor of post-processing. Trackpy *batch* function does include the sub-pixel iterative relocation method, so the output from this step is used for location result evaluation.

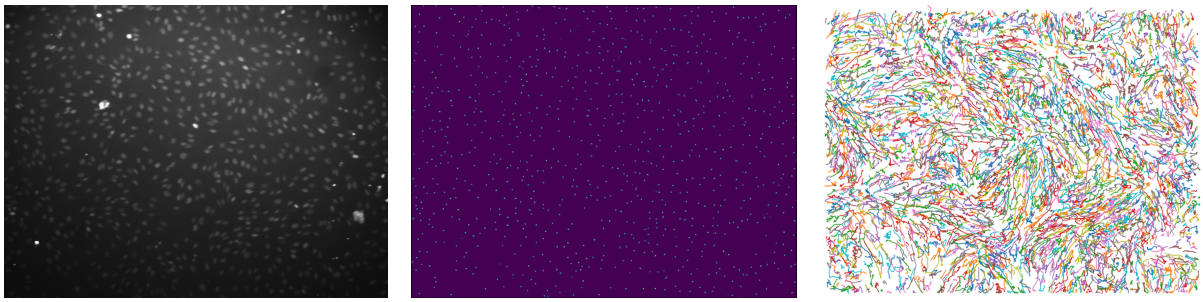

Figure 2.10: Figure showing the progression of the cell tracking pipeline from bright-field microscopy images (left), to p-maps of nucleus centroids (center), to full cell trajectories (right).

Next we link the point locations into trajectories using a search range (the function argument corresponding to the L parameter above) of 10 pixels and a memory of 5 frames. Trackpy performs efficient linking using a k-dimensional tree algorithm as implemented by Scipy [62]. While this results in a useable collection of cell trajectories, we continue to improve the trajectories using several post-processing steps. First, we use Trackpy's "stub"

filtering functionality to eliminate any particles whose trajectories exist for fewer than 10 frames. Next we interpolate the location of particles that were missing from frames but were identified as the same particle by Trackpy, which are called "memory" particles due to the Trackpy function parameter name. We use simple linear interpolation for this, using Scipy's *interp1d* method [63]. Finally, we use Trackpy's included drift-removal tools—which leverage the average cell location—to calculate, save, and remove the video drift from the trajectories [64]. We save the per-frame drift as an integer number of pixels, which allows us to correct the background images for plotting trajectories on videos.

Thus, beginning with sequences of p-maps in the form of single-channel PNGs, we obtain a DataFrame with sub-pixel locations and particle identifiers for each video in the dataset. The particle identifiers correspond to the same cell in each frame, so tracking the same particle over multiple frames provides a discrete trajectory. Figure 2.10 shows a plot of these trajectories for a non-scratch "control" video alongside samples of the images used to create the trajectories. With these we may calculate a variety of intermediate cell properties, including their angle of motion and their velocity, as well as convert trajectory data from pixels and frames to physical units such as millimeters and hours. These properties are useful for plotting and downstream analysis, but because they do not impact the tracking steps we presently forego an in-depth discussion of calculation methodology. We save these cleaned and filtered trajectories as DataFrames in storage, which allows for convenient retrieval at evaluation and analysis time.

2.4 Results

With no baseline trajectories, cell nucleus masks, or point locations for this novel dataset, we primarily focus on comparing results obtained using our custom p-map approach to the locations and trajectories acquired from the simply-preprocessed raw imagery. We first examine the improvement in unlinked cell locations from raw or contrast-corrected microscopy images to p-maps using the *chamfer distance* measure of point cloud similarity. We subsequently link the detected locations from these different input imagery dataset into trajectories using identical methodologies and post-processing steps. Finally, we quantify the error of a sample of trajectories from each of these tracking results as a means of measuring improvement.

2.4.1 Nucleus Centroid Detection

In order to examine the performance of a computer vision method, we need "true" labels for the test data. For our cell microscopy image dataset we use point locations of the cell nuclei visible in the image, as discerned by human annotators. While many state-of-the-art cell tracking performance evaluations use simulated data for testing, as it provides precise ground-truth annotations [65], the creation of synthetic data for a novel dataset is both expensive and complicated. Furthermore, we believe this approach is excessive for simple point labels, and find human annotations to be sufficiently precise.

We collect these annotations in two ways, both of which leverage human-created point labels. The first method is to have members of our research lab annotate entire frames of cells using the labelme package, an open-source Python module for image labeling [66]. The second method is to leverage Amazon Mechanical Turk, a crowdsourcing tool for distributing "microtasks" to workers that is commonly used for machine learning dataset construction [67]. Since Mechanical Turk requires small tasks, we request annotations for quarters of frames rather than for entire frames. While we consider the lab-annotated images to be more expertly labeled, Mechanical Turk allows for the rejection of improperly-labeled data, and all annotations used herein were visually inspected for accuracy.

Point Cloud Similarity Metric

Once we have collected sufficient true cell labels, we compare automatic predictions to them using the *chamfer distance* metric. Chamfer distance is a measure of point cloud similarity based on pairwise nearest-neighbor distances, which may be thought of as a symmetric variant of the sum of Euclidean errors [68]. Because the cell nucleus locations are simply points on an xy-plane, we may think of all of the locations together as a set of points in 2-space. In many imaging contexts this collection of points in space is known as a point cloud, and by defining our detections as such we may use chamfer distance to compute their similarity. Specifically, we take the sum of the Euclidean distances between each point in the predicted point cloud and its nearest neighbor in the true point cloud, do the same in the other direction for each point in the true point cloud, and then add those two sums. Thus, given two finite point clouds $X, Y \subset \mathbb{R}^n$, we define

$$d_{chamf}(X,Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|_2 + \sum_{y \in Y} \min_{x \in X} \|x - y\|_2. \tag{2.16}$$

By computing the chamfer distance between predicted and true cell nucleus point locations, we allow for a direct comparison between the various predictions.

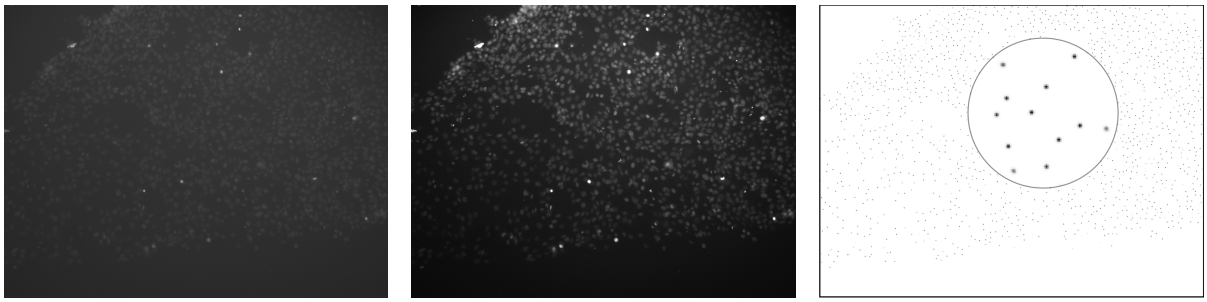

Figure 2.11: The rescaled raw image (left), sigmoid-adjusted image (center), and color-inverted p-map image (right) of the same frame of a cell microscopy video. Zoom added to p-map for visibility.

Prediction of Cell Trajectories

We have three types of input images from which we can collect cell point locations (see Figure 2.11), the raw microscopy intensity files, contrast corrected image files, and the p-maps which are output by the centroid detector. We receive files containing arrays of raw intensities, but because they are encoded as 16-bit TIFFs we first rescale them and cast them to 8-bit images, as that is the bit-depth of all other image formats and is the bit-depth required by Trackpy. Because the raw images use a very narrow range of intensities—between 3000 and 4000 values—we need not worry about losing information through this reformatting process. While we primarily use the contrast correction for visualization purposes, we also test the Crocker-Grier location algorithm on these to determine if the detection quality can be improved with a minimal, non-learned approach. To perform contrast correction, we use a pixel-wise sigmoid adjustment as made available in scikit-image's *exposure* module [69]. We use a cutoff of 0.016 and a gain of 400, which corresponds to applying

$$S(p) = \frac{1}{1 + e^{400(0.016-p)}} \tag{2.17}$$

to each pixel in the $[0, 1]$-rescaled latent image. Finally, the p-maps are saved as 8-bit PNGs and can be passed directly to Trackpy.

We use Trackpy's locating functionality to obtain a point cloud of predicted locations from each image. We do not use the locations from frames of linked trajectories, as the drift-related post-processing step obscures the precise location of the particle. Furthermore, because chamfer distance is especially sensitive to the number of predictions, we do a grid search for the "diameter" parameter and use the value that produces the number of detections which is closest to the number of true point annotations, rather than using the fixed diameter value from the full trajectory pipeline.

Centroid Detection Results

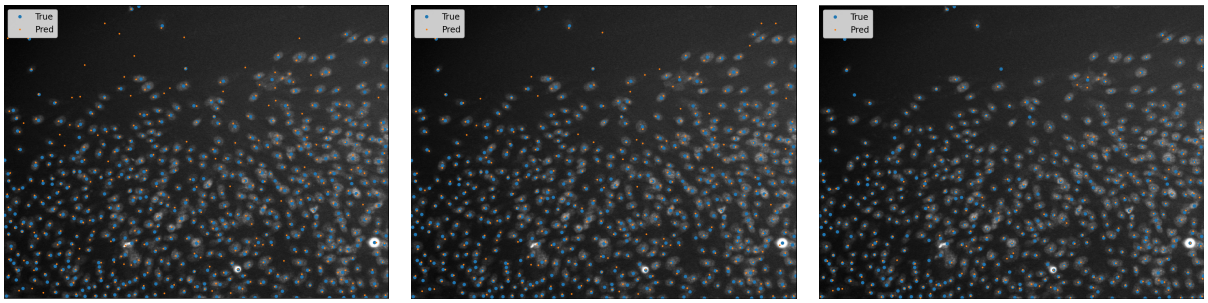

Figure 2.12: True and predicted points plotted on a background image for the scaled raw data (left), the sigmoid contrast-corrected data (center), and the centroid detector p-map (right).

Table 2.1 shows the chamfer distance from each set of predictions to the human-annotated ground-truth for both lab-annotated frames, as well as the number of detections, the corresponding diameter required to achieve that number of detections, and the percent improvement over the raw image. Table 2.2 shows the same results for the Mechanical Turk-annotated quarter frames. Each of the two test frames were labeled by two different annotators, resulting in separate true point clouds and two sets of metrics for the same predicted locations. Finally, Figure 2.12 shows the "true" annotator point labels alongside the labels predicted using each of the three input images for a single image quarter. Note that the background image does not reflect the image data that the location algorithm sees; all points are plotted on the same sigmoid-adjusted background.

Table 2.1: Chamfer distances for each of the location predictions for both lab-labeled images. The early frame (left) has 1865 true annotations and the middle frame (right) has 1579 true annotations.

Image	**# Det.**	**Diam.**	**Chamf.**	**% Diff.**
raw	1872	11	19117.511	———
sigmoid	1825	11	19097.931	-0.102%
p-map	1607	5	8701.248	-54.5%

Image	**# Det.**	**Diam.**	**Chamf.**	**% Diff.**
raw	1699	13	17138.617	———
sigmoid	1584	13	16737.591	-2.34%
p-map	1528	5	7981.142	-53.4%

Table 2.2: Chamfer distances for each of the location predictions for the Mechanical Turk annotations. The high-density quarter-frame (top) has 587 true annotations and the low-density quarter-frame (bottom) has 147 true annotations.

Annotator			# 1		# 2	
Image	**# Det.**	**Diam.**	**Chamf.**	**% Diff.**	**Chamf.**	**% Diff.**
raw	510	11	5682.025	———	5160.501	———
sigmoid	477	11	5298.928	-6.74%	4694.619	-9.03%
p-map	464	5	3401.072	-40.1%	2905.138	-43.7%

Annotator			# 1		# 2	
Image	**# Det.**	**Diam.**	**Chamf.**	**% Diff.**	**Chamf.**	**% Diff.**
raw	142	27	2705.276	———	2708.757	———
sigmoid	139	23	4001.845	+47.9%	3867.319	+42.8%
p-map	144	5	1174.075	-56.6%	1251.157	-53.8%

2.4.2 Cell Tracking Results

Because the manual tracking of cells through time is exactly the type of costly annotation process we need to avoid, we take a discrete approach to evaluating our cell trajectories. Specifically, we examine a subset of the time steps for individual experiments, counting the number of times a cell nucleus is missed, the number of times a cell is lost and then found again as a different particle, and the number of false predictions. We further look at only a cropped portion of the tile, treating it as a representative sample. Figure 2.13 shows examples of the plots used for this manual evaluation. A total of 24 cells appear in these 9 cropped frames, which are sampled every ten frames from frame 20 to frame 100. We choose this spacing as the cells move a small enough distance to still be identifiable, yet in examining 9 frames we cover a majority of the experiment's time span. While many of the cells are in each image throughout the entire range, some appear in only some frames, resulting in 182 total true detections. Using these true detections as the condition positives allows us to measure the performance with commonly-used statistics [70]. We calculate the true positive rate (TPR) by taking the ratio of these correctly identified true positives to the total number of positives. This is a useful measurement of the *sensitivity* of the approach, which describes its ability to detect all condition positives yet is unaffected by false positives. On the other hand, we can evaluate the *precision* of the test by computing what proportion of the total detections are in fact true positives, a percentage called the positive predictive value (PPV). This allows us to understand capacity of a method to successfully reject noise as non-detections. A balance of both sensitivity and precision is desirable for any approach. Furthermore, because the above statistics all relate to the detection capabilities of the algorithm, we also include the

number of times that the particle label switches as a measurement of consistent tracking. These quantitative results are included in Table 2.3.

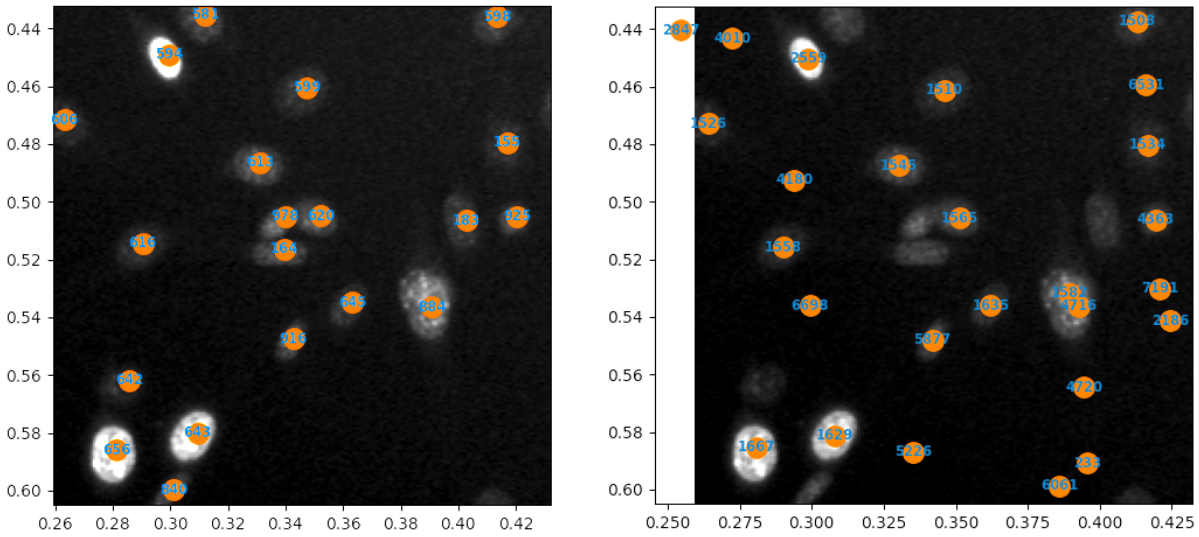

Figure 2.13: Examples of corresponding image crops used for trajectory evaluation with point annotations and particle identifiers.

Table 2.3: Results of the trajectory evaluation including the counts of true positives (TP), false negatives (FN), false positives (FP), and renumbered cells (L&F), as well as rates and proportions relating to those quantities.

Image	**TP**	**FN**	**TPR**	**FP**	**PPV**	**L&F**
raw	127	55	69.8%	110	53.6%	10
p-map	141	41	77.5%	0	100.0%	8

The p-map outperforms the raw image as input into the same Trackpy cell tracking pipeline in each of the above measures. Not only do the p-maps detect more of the actual cells in the image, as reflected by the higher number of true positives, but they also produce no false positives over the entire evaluation experiment. In contrast, the raw images produce nearly as many false positives as true positives, as indicated by the PPV of approximately 53%. Furthermore, the p-maps appear to more consistently follow the same cell over the time span, since there are fewer occasions of a cell being lost and then found again with a different particle identifier (L&F). This label-switching behavior is

exacerbated in the raw images, as 3 of the 10 switched labels previously belonged to a false positive detection, resulting in where no cell exists at all. In addition, 1 of the 8 p-map label switches belongs to a trajectory that was completely undetected by the raw image experiment until a later time. The raw image approach was not, however, completely devoid of advantages; it was able to detect one of the three cells that was left entirely undetected by the p-map approach.

2.4.3 Discussion

Based on detection point cloud comparisons, discrete trajectory-level evaluation, and visual inspection, we have shown firm evidence that our p-maps outperform raw microscopy images as input to the Crocker-Grier particle tracking algorithm. There are always, however, opportunities for the methodology to improve, primarily with how we leverage Trackpy's flexible detection and linking capabilities. In particular, we should be able to improve upon the TPR of the trajectory detections based upon the available signal in the p-maps. Figure 2.14 shows one of the labeled image crops used in the trajectory evaluation alongside the corresponding p-map.

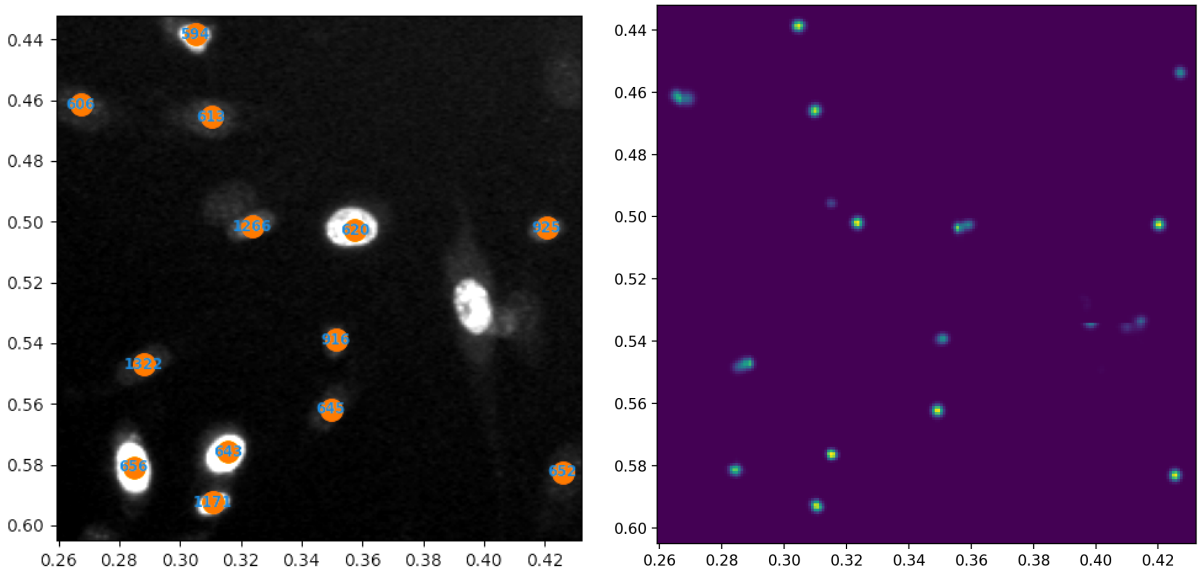

Figure 2.14: A trajectory detection evaluation frame (left) with its corresponding p-map (right). The p-map provides peaks for several cells that are missed in the trajectory data.

The p-map shows clear peaks for 4 distinct cells that are left out of the trajectory data. While these peaks are the weakest, they are still sufficient for detection by a naive

maximum filter. We cannot be sure without further investigation, but the dismissal of these p-map detections likely takes place at some point in the Crocker-Grier pipeline. Fortunately, Trackpy provides many arguments in both the *locate* and *link* functions which allow for finer tuning of the algorithm's thresholding. Whereas we have only leveraged the diameter and search radius parameters, a better TPR may be achieved by decreasing the required diameter of a p-map detection and utilizing other filtering parameters.

More generally, the balance between the TPR and the PPV may be the focus of this hyperparameter tuning. These measures may alternatively be called the *recall* and *precision* respectively, and are often in direct competition with each other [71]. In our case, by using less strict thresholds, we may be able to increase the number of true positive cell detections, but may also introduce false positive detections. This would increase the recall of the method, but may decrease the precision of the method. A rigorous optimization of the Trackpy hyperparameter arguments could aim to more appropriately balance the precision and recall. This may even be done automatically by encoding the exact information we utilize in our evaluation into cell location masks for a small number of sub-images, a relatively simple task when compared to the manual tracking of cells through each frame.

In summary, the quantitative results show that the conceptually-simple method of using CNN p-maps as input into particle tracking algorithms greatly outperforms the use of raw cell microscopy images in the same cell-tracking pipeline. Furthermore, the specific nature of the p-maps makes it easier to tune the methodology without loss of generality. This is due to the relative uniformity of the p-map detections, and the robust ability of p-maps to suppress background noise and deviations in image intensity. In this way, the p-map centroid detector acts as both a data-cleaning preprocessing step and a regularization step. As a consequence, when combined with the Crocker-Grier method for particle detection and linking, our approach provides significant performance improvements.

2.5 Trajectory Analysis

With a satisfactory set of cell trajectories available we may begin to analyze the data and uncover the mathematical properties of cell migration which will help biological researchers better model observed collective motion behaviors. There are many means of evaluating quantitative trajectory data, including the measurement of average population behaviors such as mean nearest-neighbor distances, mean speed, and total cell counts. We may also further investigate properties of individual cells, for example quantifying how far

rapidly-migrating cells move before changing direction. Finally, we may try to distinguish between physically-explainable behaviors and dynamics that appear to be random.

While a multifaceted analysis of the trajectories we have obtained is beyond the scope of this work, we aim to connect these experimental data to agent-based models by first briefly describing how we obtain velocities from the cell location, and then by examining the trajectories using a framework of biological random walks. Because the location and time data are discrete, we compute velocities for each cell at each point in time using finite differences. We then begin to quantify the observable randomness in the system by modeling it using a stochastic differential equation ABM. While an ideal model would be able to concretely explain each phenomenon contributing to cell motion and migration, we acknowledge that all models will fall short of this goal. Thus, we attempt to recover some of the dynamics we cannot model by examining how the trajectories of some cells may be modeled by randomness.

2.5.1 Velocity Calculations

Because Trackpy returns only the locations of particles in space linked through time, we must compute the velocities ourselves in order to use them in subsequent analyses. We do this using standard finite difference methods. Specifically, we use the Numpy *gradient* function and a uniform time step, resulting in centered-difference approximations for velocities within the time span and forward- and backward-difference approximations for the first and last time points respectively [72]. Thus, the cell velocity $\vec{v}^{(i)}(t_j)$ of the ith cell at time $t_j = j\Delta t$ is computed as

$$\vec{v}^{(i)}(t_j) \approx \vec{v}_j^{(i)} = \begin{cases} \frac{\vec{x}_{j+1}^{(i)} - \vec{x}_j^{(i)}}{\Delta t} & j = 0 \\ \frac{\vec{x}_j^{(i)} - \vec{x}_{j-1}^{(i)}}{\Delta t} & j = 127 \\ \frac{\vec{x}_{j+1}^{(i)} - \vec{x}_{j-1}^{(i)}}{2\Delta t} & \text{else} \end{cases}, \tag{2.18}$$

where $\vec{x}_j^{(i)}$ is the position of the ith cell at the jth time step.

Note that these simple finite difference formulas are a direct result of our assumption that the video frames are collected at precisely uniform intervals, though this is often not the case. Fortunately, the finite differences for nonuniform grid spacings can be derived using Taylor expansions [73], and the *gradient* function—provided in all recent versions of Numpy—supports arbitrary grids [72]. Thus, the same finite difference approach can be reasonably applied to any cell tracking dataset, as the requirements of smoothness are satisfied for any such biological system.

We utilize these velocities in upcoming topological evaluations of our dataset (see Chapter 5), but present the methodology here as an example of potential first steps in downstream analysis. Possible uses for these velocities include the examination of individual velocity profiles as a means of uncovering flow-like patterns of motion, or for the computation of summary statistics like average speed.

2.5.2 Random Walks

Many biological models incorporate noise in the model directly. The mathematical framework behind such models is the idea of *stochasticity*, which is the probabilistic description of randomness. By attributing some of a cell's motion to this "random" behavior, we are able to take what remains and better describe it using *deterministic* or non-random explanations. The idea of mathematically-defined stochasticity in natural systems is as old as the idea of particles themselves, as Einstein's description of random particle movement [74] was one of the first scientific accounts of what we now call *Brownian motion* [75].

We have already identified some sources of noise in our dataset, namely the unavoidable "jitter" caused by movement of the camera in microscopy data. This is an example of what we call *extrinsic noise*, which is noise in the data caused by factors outside of the system. We remove some of this noise from the data by compensating for the "drift" between frames. There are likely other sources of extrinsic noise, however, like imperfections in the cell vessel or an uneven imaging surface. Furthermore, we likely have sources of *intrinsic noise*, which are apparently random effects inherent in the system itself. The factors that cause intrinsic noise are often much less apparent as they are not factors that could be theoretically controlled unlike most extrinsic factors. To compensate for intrinsic noise is the primary reason that we include stochasticity in our biological models, but we can often capture noise from a wide variety of sources by examining the detectable distribution of the randomness in the data.

In order to include stochasticity in a model of cell motion, we first attempt to uncover how this randomness is present in our data. To do this we must first make assumptions about which components of the data are caused by noise, and which are caused by prevailing, presumably-deterministic phenomena. While some models assume that there are many quantities with varying levels of stochasticity, we will model only the location of the cell nuclei in our dataset and will assert that the cell positions we will later model are entirely stochastic. We follow the prevailing theory of Brownian motion in biology and assume that cells take *random walks*. A random walk is essentially the act of taking

a random-length step in a random direction at every point in time. These random walks are represented in continuous mathematics using *Wiener processes*, which are the generalization of Brownian motion [76]. A Wiener process is a stochastic process—a collection of random variables changing with respect to time—such that the process begins at 0 and is normally distributed with mean 0 and with variance of t at time t:

$$W(t) \sim \mathcal{N}(0, t). \tag{2.19}$$

This formulation very naturally captures the diffusive nature of random walks: as time increases, the likely range of the agent increases.

2.5.3 Stochastic Differential Equations

We connect this idea of mathematically-formalized random walks to the positions of particles in space by using the Wiener process in a basic *stochastic differential equation* (SDE) model. We assume that the position of a particle undergoing a random walk is entirely modeled by a Wiener process of a certain magnitude and that particle's initial position. Thus we have a simple SDE model for a particle in 2-dimensional space:

$$\begin{aligned} \mathrm{d}\vec{X}(t) &= \vec{\sigma}\ \mathrm{d}W \\ \vec{X}(0) &= \vec{x}_0 \end{aligned} \tag{2.20}$$

where $\vec{X}(t) \in \mathbb{R}^2$ is the position of the particle at time t and thus x_0 is its initial position, $\vec{\sigma} = [\sigma, \sigma]$ is the magnitude of the random walk, and W is a 1-dimensional Wiener process as above. Note that, while we assert that the diffusive random walk magnitude is the same in both directions by setting both components of $\vec{\sigma}$ to the same value, this need not be the case.

We make the assumption that the stochasticity is independent in each dimension, so we can simplify the process of solving the system of SDEs to solving each of the SDEs individually. We therefore only need to solve a one-dimensional, first order, linear, homogeneous SDE with a single constant term—the simplest type of differential equation—in order to uncover the relationship between our random walk parameter σ and the distribution of a particle's positions over time. Thus we solve the 1-dimensional version of (2.20):

$$\begin{aligned} \mathrm{d}X(t) &= \sigma\ \mathrm{d}W, \\ X(0) &= x_0. \end{aligned} \tag{2.21}$$

The theory of SDEs does differ from that of ordinary IVPs, but in this simplest case

the solution is identical to that of the corresponding ODE [77]. Thus the solution to (2.21) is

$$X(t) = \sigma W(t) + x_0. \tag{2.22}$$

We may extend this solution identically into multiple independent dimensions by simply sampling from this stochastic process for each direction such that

$$\vec{X}(t) = \begin{bmatrix} \sigma W(t) + x_0 \\ \sigma W(t) + y_0 \end{bmatrix} \tag{2.23}$$

where $\vec{x}_0 = [x_0, y_0]^T$ is the initial position of the particle.

For statistical modeling purposes, we need to understand the distribution of this stochastic position process. Note that, by definition of a stochastic process, $X(t)$ and $W(t)$ are random variables for any time t. By the fact that $W(t) \sim \mathcal{N}(0, t)\ \forall t > 0$, we immediately obtain

$$\begin{aligned} E\left[X(t)\right] &= E\left[\sigma W(t) + x_0\right] = \sigma E\left[W(t)\right] + x_0 = \sigma(0) + x_0 = x_0 \\ \text{Var}\left[X(t)\right] &= \text{Var}\left[\sigma W(t) + x_0\right] = \sigma^2 \text{Var}\left[W(t)\right] = \sigma^2 t \end{aligned}$$

for any value of time $t > 0$. Therefore $X(t) \sim \mathcal{N}(x_0, \sigma^2 t)$.

2.5.4 Stochasticity in Experimental Data

Now that we have a simple model describing a particle's location in a biological system in terms of measurable stochastic noise, we aim to set the σ parameter based on the observed random walk behavior of our experimental fibroblast cell data. This allows us to account for much of the noise in our data and to model random behavior using these stochastic variables. In order to directly relate the SDE parameter σ to data that can be observed, we must determine which observations in our dataset can be modeled as samples from the same stochastic process.

For this preliminary investigation we choose a single scratch assay video—position 9—from our dataset. Because we expect there to be many other forces acting on the particles that are migrating persistently, we only collect the position of a cell if its final location is within 40 microns (μm) of its initial location, a total of 97 cells. Frame 6 is chosen as the initial time, as camera movement is particularly pronounced over the first few frames. Thus we say that the cell's position in frame 6 is its initial position $\vec{x}_0$. We are essentially assuming that these cells are not moving enough to be influenced by a separate deterministic force, and instead assert that these cells move only according to

whatever biological process causes random walking. This assumption is supported by the approximate normal distribution of cell position changes, which does indeed "spread" via an increase in position over time, shown in Figure 2.15. In practice we are observing a population of multiple particles to recover σ even though our model describes only the position of a single particle. Because we assume no interactions between the particles, this can be thought of as multiple realizations of the single particle without any change in theory.

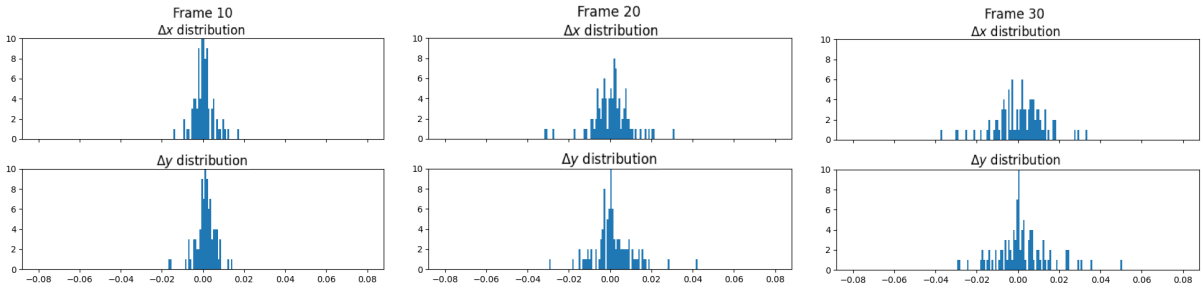

Figure 2.15: Histograms of distance differences in both directions for three separate frames in the beginning of the experiment. The histograms suggest that the positions are in fact normally distributed in each direction, and that the variance increases over time.

In order to relate experimental cell position data to the stochastic distribution of positions we may simply collect samples of where a cell moves from position x_0 at time 0 to position x at time t, obtain the variance of this sample, equate with the variance of the random variable, and then solve for σ. Thus for a collection of n cell positions collected at time t where the cell originated at x_0, $\{x_1, ..., x_n\}$, we obtain the sample variance of the changes in position $\text{Var}\left[\{x_1 - x_0, ..., x_n - x_0\}\right]$ and then estimate σ. Since we perform this analysis on the changes in position of a population of particles ($n = 97$) instead of a single particle starting from the same position x_0, we subtract the initial position of each particle individually and estimate the variance of the change in position Δx_i for the population of particles as $s^2 = \text{Var}\left[\{\Delta x_1, ..., \Delta x_n\}\right]$. From here we finally recover an estimate $\bar{\sigma}$ of the random walk magnitude σ:

$$\bar{\sigma} = \sqrt{s^2/t}. \tag{2.24}$$

This method obtains a value for σ using a set of cell positions at a single time. However, because we assume that σ is constant with respect to time, we should confirm that there is no immediately obvious evidence to the contrary. Because we have multiple time steps,

we are able to obtain an estimate for each frame of cell positions. Incidentally, each of the 97 stationary cells are present and detected in each of the 119 subsequent frames examined. Thus we collect a sequence of estimates $\{\bar{\sigma}_{t_1}, ..., \bar{\sigma}_{t_1 19}\}$ for frames 7 through 128. A sample of position difference histograms is shown in Figure 2.15. The sample variances of the position distributions illustrated in these plots are the ones used in calculating our experimental diffusion magnitude estimates $\bar{\sigma}$.

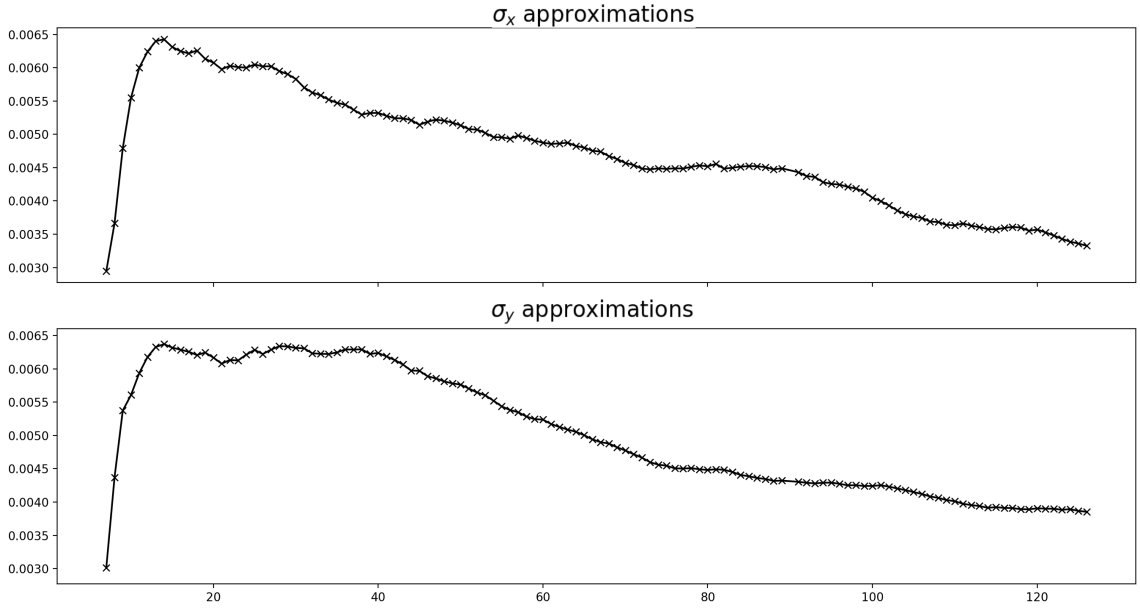

Figure 2.16: The estimates for σ in the x and y directions for each frame in scratch assay video 9.

If there were no external forces these estimates would all be equal and $\bar{\sigma}_{t_i}$ would be a constant over time. In actuality we are limited by noise not accounted for in the observations and the small number of observations. The plot of all estimates $\bar{\sigma}$ is shown in Figure 2.16. The quality of the estimations decreases over time because a small number of samples does a progressively worse job of approximating the normality of the distribution as the variance increases. In contrast, the initial estimates are likely limited by imperceptibly small steps from the initial position, resulting again in poor approximations of the true value. Histograms demonstrating both issues are shown in Figure 2.17. The best visual similarity to a normal distribution of position differences occurs between frames 20 and 30 (Figure 2.15). Therefore we assert that the near-constant value of $\bar{\sigma} \approx 0.006$ over that range of frames is the best choice of the true random walk magnitude, so $\sigma = 0.006 \ \frac{\text{mm}}{\sqrt{\text{hr}}}$.

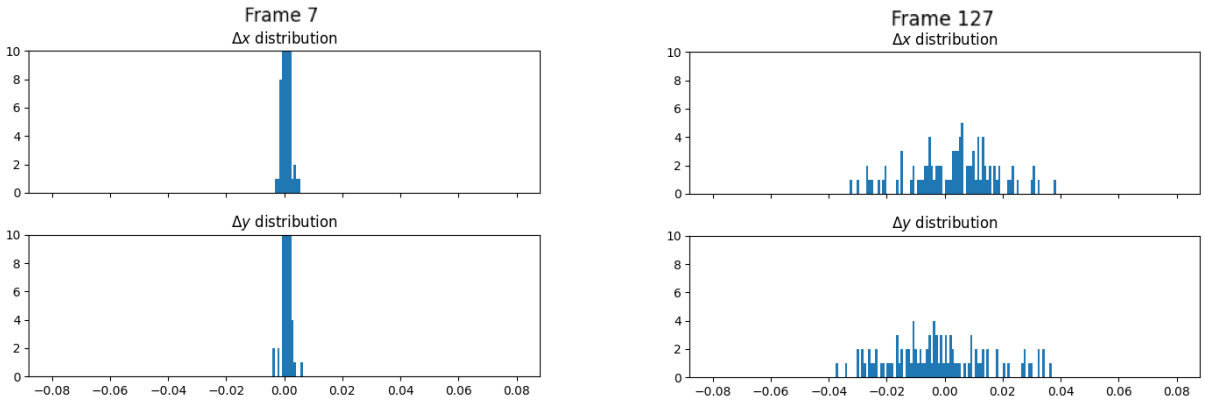

Figure 2.17: Histograms of distance differences showing too narrow a distribution in early frames (left) and too few samples to accurately model a normal distribution in late frames (right).

2.5.5 Discussion

This type of investigation is only one way to analyze the stochasticity in experimental data. We may use different mathematical models, agent-based or otherwise, or even attempt to fit this data to a model and subsequently quantify the uncertainty as a means of examining the random noise. However, as an initial investigation, we have shown the worth of accurate experimental trajectories as a means of understanding cell migration via velocity calculations and evaluation of random walk behavior. By bringing collective cell movement into the realm of point clouds and quantitative variables, we enable the rigorous mathematical investigation of our experimental data.

2.6 Conclusions

2.6.1 Contributions

Our primary contribution to the cell tracking corpus is the application of a p-map approach to a single-image training dataset, and the use of p-maps to perform cell tracking using the Crocker-Grier algorithm. While the p-map approach is well-documented as a method for locating cell nuclei in a variety of microscopy datasets, nearly all of these implementations use at least a modest-sized training dataset, transfer learning from a publicly-available microscopy dataset, or both [52]. Our technique of training our fully-convolutional neural network from scratch on tiles from a single image appears to be practically novel. While we do acknowledge the limitations of this approach in terms of dataset generalizability, it represents a means for improving automatic cell nucleus detection with a minimal set of

labeled training images and an alternative to the aforementioned one-shot methods. Our successes suggest that computer vision techniques can be applied to even the most data-limited problems for cell localization. Other features of our centroid detector represent incremental changes from previously-published approaches. These include an especially-deep UNet-type architecture, where other p-map implementations have used similar fully-convolutional networks with fewer weights and layers [52].

The the best of our knowledge, we are the first to employ a p-map cell nucleus detector as the first step in a cell-tracking pipeline. While the Crocker-Grier algorithm is rather popular for tracking colloidal particles, its use for tracking general populations of cells is less prevalent [78]. In particular, it is built around the assumption that particles are approximately spherical, which does not hold for most cell tracking purposes. However, by designing a neural network output format that represents cell centroids as Gaussians, we recover this premise and the Crocker-Grier approach becomes a good fit for cell trajectory linking. While other approaches have implemented expansions to the p-map approach, in some cases even developing full cell segmentation functionality [79], it does not appear that p-maps have heretofore been used for obtaining cell trajectories through time.

Finally, we have taken significant steps toward a useful mathematical analysis of this novel fibroblast migration dataset. While the focus of this work is not to answer specific biological questions, we equip biological researchers with the tools to not only draw conclusions from this specific dataset of 42 scratch assay videos, but also to analyze future similar datasets, perhaps without any need for retraining. We have shown the relative ease of acquiring quantitative summaries of the cell motion by investigating the random walk behavior of stationary cells in a single video, and anticipate that other investigators will employ similar approaches. This also gives us a way to generate large time-varying point cloud datasets from experimental data, a task that would be impossible with manual annotations. Not only does this provide immediate quantitative data, but also enables our subsequent topological data analysis approach.

2.6.2 Future Work

While the motivation for this work is to acquire useful trajectories from our experimental scratch assay data for the sake of analysis, we believe this approach has merit for a wide range of cell tracking applications. Our approach does indeed depend on relative cell nucleus uniformity for single-image network training and sufficiently-small cell step sizes for nearest neighbor linking, but these requirements may be reasonable for many types of cell microscopy video data. Furthermore, this approach is not limited to single-channel

image data, and can be applied to color imagery and potentially even non-visual imagery.

We also believe this p-map approach represents an excellent starting point for the development of end-to-end-learned cell tracking pipelines. As the capabilities of deep neural networks continue to surpass traditional approaches and even other machine learning approaches, the modest nearest-neighbor method for linking trajectories may be best replaced by a technique that is based on the movement of the observed data. Recent cell tracking research utilizes novel methods like the so-called "Siamese network" for an entirely-learned approach [80], and we believe this same idea could be applied to p-map processed imagery in order to reduce the annotation demands. Overall, the advantage of requiring only inexpensive point labels for training makes the p-map approach ideal for prototyping computer vision methods on novel, unannotated biological time-series datasets.

Chapter 3

Background Material: Persistent Homology and Topological Data Analysis

3.1 Introduction

The primary aim of this dissertation is to describe and categorize the collective motion present in variety of datasets using *topological data analysis* techniques. Because these methods use topological concepts not often discussed in conjunction with applied data science, we present important preliminaries from the field of algebraic topology, including the mathematical notions of simplicial complexes, homology, and persistent homology. Thus this brief chapter does not present any new results, but simply provides a detailed overview of the topology needed to understand the ensuing data analysis methods.

By defining a point cloud dataset as a finite metric space we are able to apply notions of distance, length, and shape. This is exactly the motivation behind a growing field of data investigation called *topological data analysis* (TDA). TDA combines concepts from applied algebraic topology, computational geometry, and descriptive statistics to provide data-driven methods for characterizing the underlying *shape* of data [81].

TDA is a rapidly-expanding area of research, and as such new methods are being constantly developed and introduced. However the basis of all TDA is to turn a point cloud into a topological object whose properties can be studied, and then to summarize those properties in ways that are useful for data analysis. This approach to data analysis has clear applications to biological research, as it is directly applicable to graphs that appear in phylogenetics, connectomics, and the study of protein structures, as well as to

biological imagery analysis and—most pertinently—the movement and migration of all varieties of cells [82].

The TDA process is thus a multi-step pipeline, beginning with a point cloud and ending with qualitative and quantitative representations of the shape of the dataset. The introductory article by Frédéric Chazal and Bertrand Michel [81] provides an excellent synopsis of this pipeline, as well as relevant background material, examples, and applications. Research regarding the application of TDA usually revolves around how to best use these topological representations to accomplish an objective.

Our interest in TDA relates to its ability to quantify behaviors at a wide variety of scales, a quality that allows us to both estimate the parameters of a mathematical model at an infinitesimal level and observe large-scale collective motion phenotypes in experimental data. Specifically, because TDA has a unique ability to concisely summarize the shape of a point cloud both locally and globally, it is ideal for models that result in multi-scale behaviors. Since ABMs epitomize this behavior, we thus relate the output of our cell migration models to the topology of its attributes via TDA of its point cloud representation.

The tool that topological data analysts most often wield is the computation of the *persistent homology* (PH). By quantifying the PH of a point cloud we obtain a useful summary of its shape on a range of scales, parameterized by the spatial resolution. Most relevantly, we may use a point cloud's persistent homology as a basis for comparison to other point clouds. The full PH pipeline consists of forming a *simplicial complex* from a cloud of points, building a *filtration* of complexes, measuring the *homology* of each of those complexes, and finally summarizing how the homology changes over the sequence of complexes in the filtration. Each of these concepts is introduced below, followed by an explanation of how the pipeline is implemented in software and how the summary measures are used in practice via *crocker plots*.

3.2 Simplicial Homology

Homology is a fundamental topological idea which formalizes the congruence of manifolds based on the number of holes which they enclose. While the primary concern of algebraic topologists is the homology of manifolds, it is more relevant to TDA to introduce only the notion of *simplicial homology* [83].

In simplicial homology we compute the homology of a *simplicial complex*—a collection of fundamental topological building blocks called *k-simplexes*. A k-simplex is the simplest k-dimensional geometric polytope, which can be thought of as the generalization of a

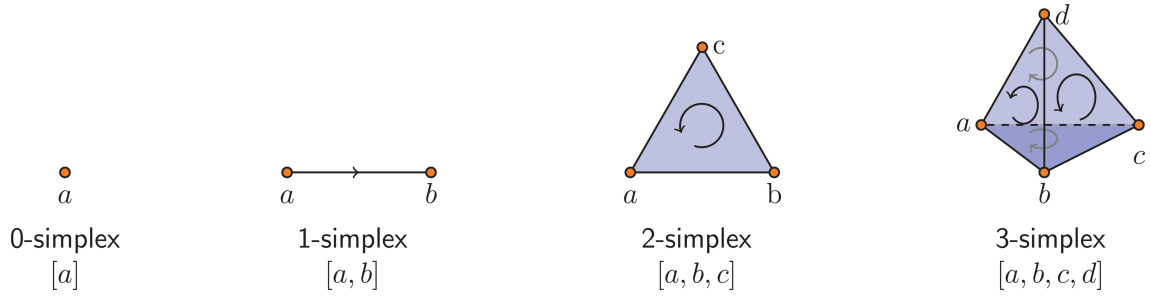

Figure 3.1: An illustration of the first 4 simplexes with vertex representations [84].

tetrahedron [85]. In other words, a k-simplex is the combination of k distinct points into edges, triangles, tetrahedra, and higher-dimensional generalized tetrahedra in its corresponding k-dimensional space. The first few simplexes are illustrated in Figure 3.1. We call any of the $(k-1)$-sub-simplexes that form a k-simplex a *face* of that simplex. Any collection $\mathcal{K}$ of simplexes where all the faces of a simplex in $\mathcal{K}$ are also in $\mathcal{K}$ and where the intersection of any two simplexes of $\mathcal{K}$ is a face of both forms a simplicial complex $\mathcal{K}$ [86]. A k-simplex σ may be described by its *vertices* v_i, which are the k 0-simplexes that span the k-simplex. This results in the common notation $\sigma = [v_0, ..., v_k]$, demonstrated in Figure 3.1.

The idea of simplicial homology is to quantify the number of holes that a simplicial complex encloses in any dimension. To formalize this notion and to eliminate any ambiguities in what may be regarded as a "hole", homology is defined in terms of *chains* of simplexes, the *boundary* of a simplex, and subsets of those chains called *cycles*. We now introduce each of these concepts in turn, leading to a purely algebraic formulation of homology. Once again, the treatment of simplicial homology provided in the Chazal article [81] is highly recommended for those interested in TDA who lack a formal background in topology.

Due to the algebraic nature of topology, we collect simplexes using addition, a binary operation. This contrasts with the elementary geometric concept of a collection of points and edges as a set (either ordered or unordered). Thus, given a simplicial complex $\mathcal{K}$, a collection k-simplexes in $\mathcal{K}$ is a sum called a *k-chain*. In particular, any k-chain can be written as

$$c = \sum_{i=1}^{p} n_i \sigma_i, \tag{3.1}$$

where $\{\sigma_1, ..., \sigma_p\}$ is the set of k-simplexes of $\mathcal{K}$ and $n_i \in -1, 0, 1$ [87]. The space of k-chains of $\mathcal{K}$ is the span of all k-simplexes in $\mathcal{K}$, denoted $C_k(\mathcal{K})$. Because we use addition to make chains from simplexes chains themselves can also be summed, and $C_k(\mathcal{K})$ naturally

forms a group under addition.

To compute the homology of a simplicial complex we must first define the notion of a *boundary*. Just as we intuitively conceive of a triangle's boundary as consisting of its three edges put together, a k-simplex is bounded by a chain of $(k-1)$-simplexes. We say that—given a k-simplex represented by its vertices $\sigma = [v_0, ..., v_k]$—the *boundary operator* ∂_k yields the boundary, a $(k-1)$-chain defined by

$$\partial_k(\sigma) = \sum_{i=0}^{k} (-1)^i [v_0, ..., \hat{v}_i, ..., v_k], \tag{3.2}$$

where $[v_0, ..., \hat{v}_i, ..., v_k]$ is the $(k-1)$-simplex spanned by each vertex **besides** v_i. Since $C_k(\mathcal{K})$ is a space whose basis is the k-simplexes of $\mathcal{K}$, we may naturally apply the boundary operator to chains as well. Thus, given $c \in C_k(\mathcal{K})$, $\partial_k(c)$ is still a $(k-1)$-chain and furthermore ∂_k is a linear operator from $C_k(\mathcal{K})$ onto $C_{k-1}(\mathcal{K})$.

Finally, a *cycle* is a chain with zero boundary, i.e., a chain $c \in C_k(\mathcal{K})$ such that $\partial_k(c) = 0$. A cycle may be simplistically thought of as as a chain that begins and ends at the same face. More formally, the space of k-cycles of $\mathcal{K}$ is the *kernel* of the boundary operator, i.e., $Z_k(\mathcal{K}) = \operatorname{Ker} \partial_k = \{c \in C_k(\mathcal{K}) \mid \partial_k(c) = 0\}$.

Taking the space of k-cycles along with the space of k-boundaries, denoted by $B_k(\mathcal{K}) = \operatorname{Im} \partial_{k+1} = \{c \in C_k(\mathcal{K}) \mid \exists d \in C_{k+1}(\mathcal{K}) \ni \partial_{k+1}(d) = c\}$, we define the kth *homology group* of the simplicial complex $\mathcal{K}$ as the quotient space

$$H_k(\mathcal{K}) = Z_k(\mathcal{K}) / B_k(\mathcal{K}). \tag{3.3}$$

Intuitively, this homology group quantifies how many k-cycles are not k-boundaries. More accurately, each element of $H_k(\mathcal{K})$ is an equivalence class of **all** cycles that are *homologous*. The elements of these equivalence classes are cycles that differ by no more than a boundary, and thus all these cycles enclose the same topological feautres. Therefore each of these classes have a correspondence to a specific cycle that is not a boundary. This is what eliminates ambiguity in the counting of holes in a simplicial complex; there may be many k-cycles that "enclose" one particular k-dimensional hole, but since they all belong to the same equivalence class the number of such classes provides a count of holes.

We consider a single triangle (2-simplex) with vertices v_0, v_1, v_2 as pictured in Figure 3.2 as a simple example of a simplicial complex. The components of this simplicial complex are the 2-simplex, three 1-simplexes (edges), and three 0-simplexes (vertices), so we let $\mathcal{A} = \{[v_0, v_1, v_2], [v_0, v_1], [v_0, v_2], [v_1, v_2], [v_0], [v_1], [v_2]\}$. As shown in the figure, the boundary of the 2-simplex is a chain of 1-simplexes. Furthermore, the boundary of that

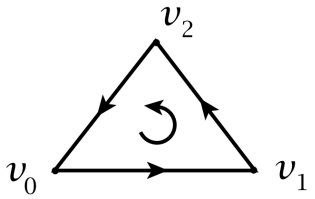

Figure 3.2: A 2-simplex and its boundary [88].

chain is 0:

$$\begin{aligned}\partial_1([v_1, v_2] - [v_0, v_2] + [v_0, v_1]) &= [v_2] - [v_1] - ([v_2] - [v_0]) + [v_1] - [v_0] \\ &= [v_2] - [v_1] - [v_2] + [v_0] + [v_1] - [v_0] \\ &= \cancel{[v_2]} - \cancel{[v_1]} - \cancel{[v_2]} + \cancel{[v_0]} + \cancel{[v_1]} - \cancel{[v_0]} = 0.\end{aligned} \tag{3.4}$$

This means that $\partial_2([v_0, v_1, v_2])$ is a cycle. In fact, every boundary is a cycle. Formally, $\partial_{k-1}\partial_k(c) = 0$, an easily provable fact and a foundational theorem of simplicial homology. Though other 1-chains exist in this complex, it is easy to see that there are no other 1-cycles. Furthermore, since there is only a single 2-simplex there are no other 1-boundaries. Thus we can show that there are no 1-dimensional holes in this simplicial complex by computing the 1st homology group:

$$\begin{aligned}H_1(\mathcal{A}) &= Z_1(\mathcal{A}) / B_1(\mathcal{A}) \\ &= \{[v_1, v_2] - [v_0, v_2] + [v_0, v_1]\} / \{[v_1, v_2] - [v_0, v_2] + [v_0, v_1]\} \\ &\cong \{0\}.\end{aligned} \tag{3.5}$$

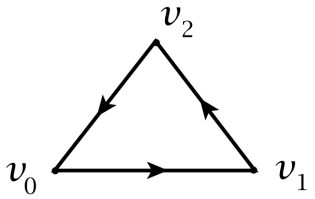

Figure 3.3: A triangular 1-cycle.

We now consider a second simplicial complex $\mathcal{B}$, formed by removing the 2-simplex but leaving all the other simplexes (Figure 3.3). We see that the only 1-cycle is no longer a boundary, as there is no 2-simplex to bound. That cycle now encloses a hole and we are

left with a non-trivial 1st homology group:

$$H_1(\mathcal{B}) = Z_1(\mathcal{B})/B_1(\mathcal{B}) = \{[v_1, v_2] - [v_0, v_2] + [v_0, v_1]\}/\{0\} \cong \mathbb{Z}. \tag{3.6}$$

Homology being a construction of algebraic objects results in the natural setting of groups and isomorphisms. However, since we are interested in simply quantifying the number of holes, we deal directly with the **rank** of the kth homology group of the simplicial complex, $\beta_k(\mathcal{K}) = \dim H_k(\mathcal{K})$. This positive integer β_k, called the kth *Betti number*, gives a count of the k-dimensional holes enclosed by the simplicial complex [89]. These Betti numbers are the main quantities with which TDA is concerned, as they concretely represent the shape of a complex in terms of the holes they contain.

While this short summary provides all the algebraic topology one needs to understand how the Betti numbers of a simplicial complex are computed via its homology, in-depth treatments and the accompanying theorems can be found in Hatcher's *Algebraic Topology* [90]. An applied approach to the treatment of homology and how it relates to TDA can be found in Robert Ghrist's *Elementary Applied Topology* [91].

3.3 Turning a Point Cloud into a Simplicial Complex

We now have a concrete method for summarizing the topology of simplicial complexes via homology and Betti numbers. Yet given a collection of discrete points in some n-dimensional space—a point cloud—we are not given a simplicial complex. We must first connect this set of 0-simplexes (points) into a complex of higher-dimensional k-simplexes before we are able to draw any conclusions about the topology of the data. The most natural and most frequently-utilized method for building a complex from points is to form connections between neighboring points. Since we conceive of a point cloud as a finite space endowed with a metric, we already have the measure of closeness needed to describe which points are neighbors.

Let (X, d) be the finite metric space given by a point cloud dataset X with metric d. Given a *proximity parameter* $\varepsilon > 0$, we may construct the *Čech complex* by connecting all points whose $\varepsilon/2$-balls have a common point of intersection [92]. This method of forming a complex is not only conceptually straightforward, but is well studied from a topological perspective and comes with guarantees about its homotopy via the "nerve theorem" [92]. The Čech complex, however, is especially expensive to compute, as it relies on the absolute locations of points in space. Therefore, although the Čech complex is the most natural simplicial complex for point cloud data, TDA most often leverages the

Vietoris-Rips (VR) *complex*.

The VR complex of (X, d) is constructed by connecting any collection of $k+1$ points that are pairwise within ε of each other into a k-simplex [93]. Put another way, we draw the same $\varepsilon/2$-balls used in the construction of the Čech complex, connect balls that are overlapping, then form all points which are pairwise-connected into simplexes. This construction allows the VR complex to be constructed using only the distances between points, which makes computation more efficient. Furthermore, the entire VR complex can be stored as a graph, making it ideal for applied algorithmic approaches. The proximity parameter used to build a complex defines the *spatial resolution* of the complex, with smaller values of ε resulting in *finer* spatial resolutions and larger values resulting in *coarser* ones. Figure 3.4 in the next section shows how VR complexes of a 2D point cloud may be constructed at a variety of spatial resolutions, but note that this same exact construction applies to any dimensional space.

3.4 Persistent Homology of a Filtration of VR Complexes

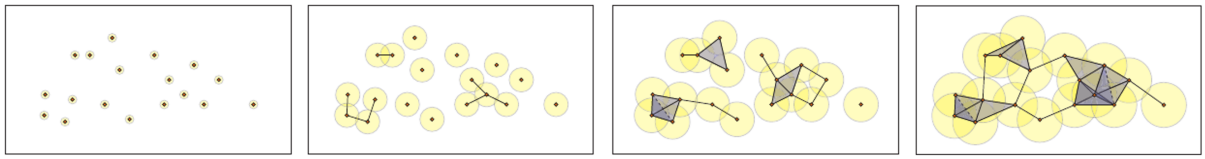

Figure 3.4: Sample filtration of a VR complex for a 2D point cloud. The $\varepsilon/2$-balls are shown to illustrate how the simplexes are formed [94].

The Vietoris-Rips complexes formed by varying the proximity parameter ε have the convenient property that any complex constructed at a finer spatial resolution is a *subcomplex* of any complex constructed at a coarser spatial resolution. Specifically, given that $VR_\varepsilon(X)$ is the VR complex of a point cloud X with proximity parameter ε, $VR_{\varepsilon'}(X) \subseteq VR_\varepsilon(X)$ if $\varepsilon' \leq \varepsilon$. Taking a sequence of proximity parameters $T = (\varepsilon_0, \varepsilon_1, ..., \varepsilon_m)$, we can then define an entire nested family of VR complexes $(VR_\varepsilon(X))_{\varepsilon \in T}$, called the *filtration* of the coarsest complex $VR_{\varepsilon_m}(X)$ [81]. The values of the proximity parameters can theoretically continue on to ∞. However, since we deal only with finite metric spaces, once the proximity parameter has surpassed the diameter of the metric

space each subsequent complex will be equal to that "maximal" complex. Thus, both conceptually and computationally, we let the coarsest complex in our family be this maximal complex such that $\varepsilon_m = \text{diam}(X) = \sup\{d(x, y) : x, y \in X\}$.

The motivation behind *persistent homology* is quantifying how the homology changes between each of the component complexes in a filtration. Nominally, we are tracking how topological holes "persist" throughout a range of spatial resolutions.

Given a filtration of a VR complex, we can keep track of how the homology groups change as the spatial resolution varies. In other words, we see which proximity parameter values ε cause new topological features to form. We refer to the first value of ε which results in the creation of a topological feature as that features's *birth* value, and the last value in the sequence for which the feature exists as its *death* value. This results in a *lifespan* interval for each topological feature in the filtration. Plotting all lifespan intervals over the entire range of spatial resolutions, grouped by which homology group the feature belongs to (the feature's *dimension*), gives a TDA visualization known as a *barcode*. This barcode shows the birth and death of each topological feature that exists in any subcomplex of the filtration, and thus summarizes through how many of those subcomplexes it persists.

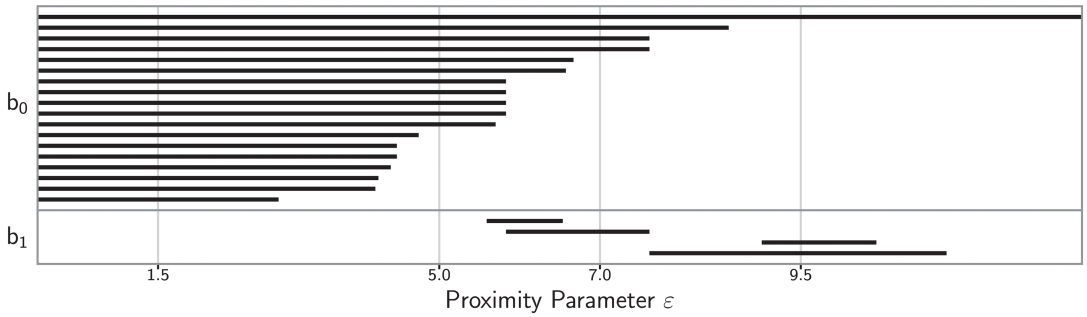

Figure 3.5: The persistence barcode of the point cloud in Figure 3.4. The b_0 bars correspond to members of the 0th homology group (connected components), and the b_1 bars correspond to members of the 1st homology group (flat holes) [94].

Barcodes are useful not only as tools for visualizing the persistent homology of a point cloud, but also as an intermediate step in the convenient computation of the Betti numbers with which we are most concerned. For any given value of ε the number of bars for the kth homology group gives the kth Betti number. So for the barcode in Figure 3.5, the 0th Betti number at $\varepsilon = 9.5$ is 1 and the 1st Betti number at $\varepsilon = 9.5$ is 2. Since there are TDA software packages that can very efficiently compute the barcode of a point

cloud, we will compute Betti numbers in exactly this way.

There exist other tools for both the visualization and concise presentation of a point cloud's persistent homology. Many of these even include further guarantees regarding stability over time and over spatial resolution. Some are direct implementations of the lifespan intervals that comprise barcodes, and others introduce novel methods for using the persistence information of a filtration. Most of these are beyond the scope of this work, and we instead focus on the *crocker plot* and its corresponding vectorized form, introduced herein.

3.5 Betti Numbers and Crockers

Several unique TDA tools exist which leverage the homology of a point cloud as summarized by its Betti numbers. Because each simplicial complex in a filtration may have a different kth Betti number, the complete persistent homology of the filtration is often summarized by considering the Betti number of each complex in the filtration as a function of the proximity parameter (ε). The graph of this function, referred to as the kth *Betti curve* of the filtration, provides a picture of how the homology changes as the proximity parameter grows [95]. However, summarizing the topology of a sequence of point clouds over time is a non-trivial problem, as the topological guarantees of filtrations break down over independent time steps.

One approach to these *time-varying* or *dynamic* point cloud datasets is to simply compute the Betti curves for each discrete time step, and then concatenate them into a single matrix which summarizes the homology of the entire time-series dataset. This is the idea behind the "Contour Realization Of Computed K-dimensional hole Evolution in the Rips complex" or *crocker plot*, created by Chad Topaz and colleagues [97]. It shows changes in Betti numbers over time by plotting the level sets or *contours* of the concatenated Betti curves. While introduced as a simple way to visualize the homology of dynamic point cloud data for the sake of understanding the topology of biological models (Figure 3.6), it has proven useful as an uncomplicated method of summarizing time-varying Betti number behavior.

Further research by the designers of the crocker plot suggests that the matrix form of concatenated Betti numbers, which they also call crockers, and which we refer to as a *crocker matrix* to reduce ambiguity, may be used as input into machine learning algorithms [98]. This prior work shows that crockers can be a useful tool for selection of aphid locomotion models, and highlights other important characteristics of the crocker matrix. They illustrate that, despite lacking the same topological guarantees, crockers

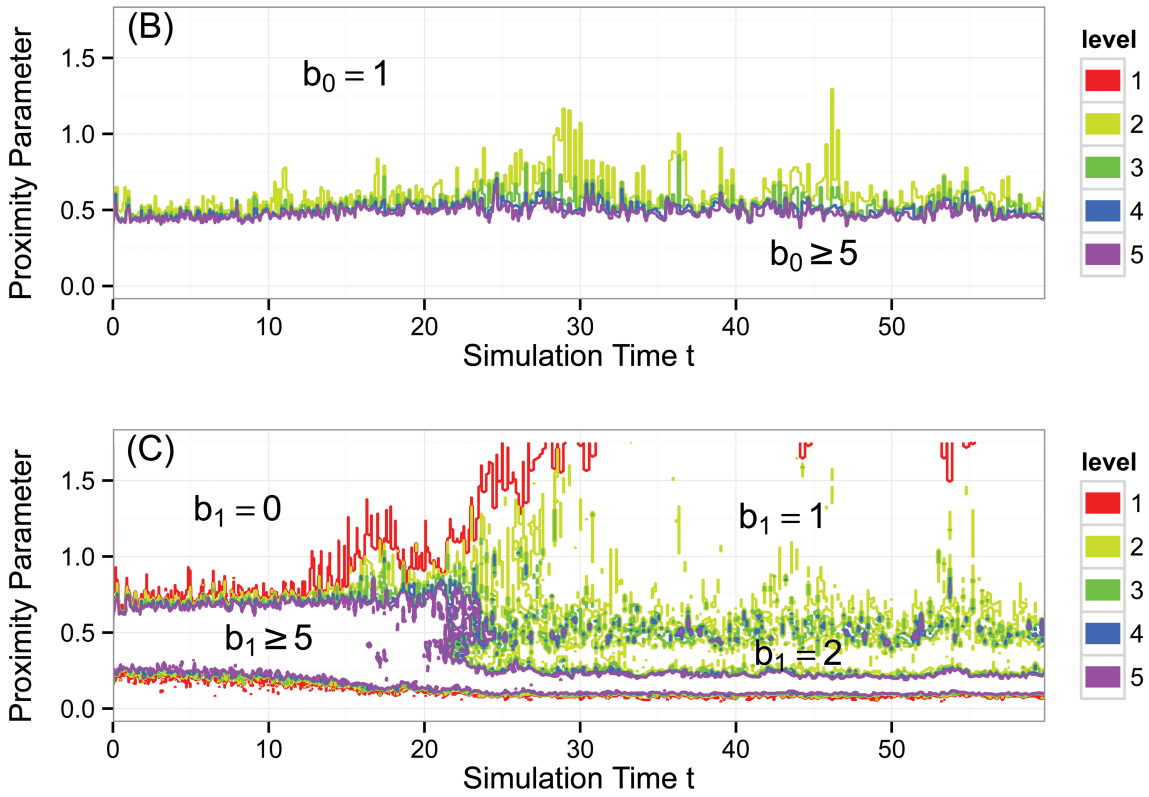

Figure 3.6: A crocker plot visualization of simulated D'Orsogna model data showing contour plots of the 0th Betti numbers (b_0, top) and 1st Betti numbers (b_1, bottom) [96].

are robust to noise with respect to time. This is especially relevant for stochastic models that introduce random noise into simulated data. The study demonstrates that the computation of a point cloud's persistent homology is application agnostic and thus requires no *a priori* knowledge of either the data or the model. They also highlight the ability of crockers to take into account a wide variety of relevant biological information, such as aphid proximity and population density. Overall, this research exemplifies the relevance of crockers for dynamical biological model analysis, and expands the use of crockers beyond scientific visualization and into the realm of TDA summary statistics via crocker matrices.

3.6 TDA Computation Software

While the computation of the simplest homology groups H_0 and H_1 is uncomplicated for simple examples, it is not trivial to compute these groups for large point clouds comprised of many data points. These computational challenges are largely due to the fact that

a relatively small point cloud may result in a filtration consisting of tens or hundreds of thousands of times more simplexes. For example, a 2000-point sampling from the well-known Stanford Dragon reconstruction results in a filtration consisting of over 1 billion simplexes [82].

The first step in computing the persistence barcodes of a filtration of simplicial complexes is to form the *boundary matrix* which enumerates the appearance of each of the simplexes in the filtration relative to other simplexes. Because the number of simplexes in a filtration is often many orders of magnitude greater than its original point cloud, this matrix is often prohibitively large, and it may not be possible to explicitly store it in working memory, which creates problems for straight-forward processing. Furthermore, the worst-case complexity of the standard method of processing the boundary matrix into the persistence barcodes is cubic in the (potentially massive) number of simplexes [82]. Thus a naive approximation of the computation time for a billion-simplex filtration would approach tens of minutes, even by modern processing speed standards.

Due to these difficulties, the past decade has seen a strong focus on the development of novel optimizations for the computation of persistence barcodes as well as the implementation of such optimizations in software libraries. *Ripser* has been the most successful of these software packages to date. It is a C++ library that utilizes many state-of-the-art optimizations for calculating persistent homology to drastically reduce both memory demands and computation time [99]. The in-depth discussion of these optimizations is beyond the scope of this work, but is thoroughly covered in both the Ripser technical writeup [99] and the PH software evaluation article by Otter and colleagues [82].

Ripser's mathematical algorithmic approach utilizes topological facts we have already introduced in the computation of the VR complexes, like the theorem that all boundaries are cycles and the notion of a maximal complex. It also implements the computation of the complex's *cohomology*, which has been shown to greatly reduce algorithmic complexity. The cohomology groups of a simplicial complex are the categorical *duals* of the homology groups, but may also be thought of as groups arising from *cochains*, which are dual to chains in the standard algebraic sense [100]. Specifically, cochains are linear maps on the space of chains, making the group of cochains the dual of the group of chains [101]. Because cohomology groups provide the same measure of a VR filtration's persistence as homology groups [102], cohomology computations are used to provide the same persistence barcodes while using fewer resources. Ripser also uses ideas from discrete Morse theory to skip over the explicit construction of many columns of the *coboundary matrix*—the cohomology equivalent of the boundary matrix. Lastly, it uses a novel ordering of the simplexes based on Knuth combinatorial numbering and uses this ordering of simplexes

to avoid storing the entire coboundary matrix, which has the effect of greatly reducing memory demands [99]. Together, these optimizations result in memory savings and speedups of at least an order of magnitude over other published TDA software libraries for a variety of benchmark tasks [82].

In an attempt to make TDA more accessible to researchers, the Scikit-TDA library furnishes a Python port of Ripser among a suite of other tools for TDA and visualization [103]. By providing Python bindings for the underlying C++ codebase, Scikit-TDA maintains nearly all of the speedups and advantages of Ripser while being extensible for use in data science pipelines. We use this Scikit-TDA module for all persistent homology computations in the following sections.

Chapter 4

Estimating Agent-Based Model Parameters using Topological Data Analysis

4.1 Introduction

The primary novelty of this research lies in the investigation of how the topology of point clouds may be used for fitting agent-based models (ABMs) to those datasets. Using a rigorous mathematical frame of reference that views the task of *parameter estimation* as a specific subset of *inverse problems*, we are able to develop new approaches for selecting continuous parameter values that best fit a given dataset. We achieve this by using operators on the space of point clouds as objective functions for minimizing the difference between ABM output and a ground-truth dataset. We are most interested in operators based on topological data analysis (TDA), as they can efficiently encode the "shape" of a point cloud dataset on multiple scales.

4.1.1 Parameter Estimation

Agent-based models present unique challenges for model calibration and parameter estimation compared to typical continuum models. The cause of this is most directly captured by the difference in scales between the observable phenomena of the system and the inputs to the model. Whereas the parameters deal with micro-scale behaviors of the agents, the most perceptible outcomes of model simulations are at the population level. This is where the inherent difficulty of developing macro-scale continuum models can be seen as a compromise; the output quantities are usually on the same physical

scale as the model parameters, making the relationship between them for the sake of parameter estimation much simpler. For example, the seminal Fisher–KPP equation uses the population-level rate of diffusion across a boundary to parameterize the diffusivity of the system as a whole [104].

Due to these difficulties, there is often a disconnect between the development and presentation of ABMs and the investigation of parameter estimation. A 2013 survey of peer-reviewed ABM papers showed that only about one quarter of these studies included any treatment of parameter estimation for their proposed models [105]. This lack of parameter investigation likely stems from the combination of a large number of model parameters and the high computational costs which make most parameter search algorithms prohibitively expensive. However, as computational efficiency has increased dramatically over the past decade, so has the number of methods for estimating ABM parameters.

Some studies have used surrogate models for parameter estimation to avoid evaluating the ABM for each iteration of a parameter estimation algorithm [106][107]. This approach, while increasingly powerful due to advances in machine-learned surrogates, will always lose some of the information that may be ascertained from genuine model simulations. The primary concern of surrogate models is the disposal of intermediate information, such as the individual contributions of multiple agents to interactive forces in biological flocking systems. When simulating the full ABM, parameter estimation can be based on either comparing the similarity of the outputs directly at the micro-scale agent level or by developing a summary for the model which attempts to capture the macro-scale behaviors or a *multi-scale* combination of behaviors. Local comparisons can be straightforward, enabled by the obvious ability to compare points spatially using distance metrics when data is presented as a point cloud. This however is especially ineffective for stochastic ABMs, where noise is most often prevalent at smaller scales. The alternative of using a summary measure of behavior for the simulated data has proven effective, but requires expert knowledge and deep understanding of the model output. Many methods also require the calculation of additional attributes besides what the model yields, such as average population behaviors, which also reintroduce some of the limitations of continuum models.

Due to these limitations, state-of-the-art parameter estimation studies leverage computationally-tractable aggregates that measure the desired multi-scale behaviors of ABM simulation data. This method of developing summaries is the focus of several recent papers, and is the main focus of the ensuing analysis. The lab of Dr. Matthew J. Simpson has used a discrete approximation of the *density profile* of an image to much

avail in the study of ABMs for cell diffusion in scratch assay experiments. Density profiles are calculated by splitting an image up into a fixed number of rectangles, counting the number of cells in each rectangle, and dividing by the area of the rectangle to obtain an approximation of the number of cells per unit area as a function of the position in any one direction across the image [15]. This is only one example of a function that may be used to summarize emergent behavior in the field of cellular migration. Others leverage raw counts of cells, particularly in models that involve proliferation [108], and easily measured cell population properties such as radius and area [109].

In summary, research into parameter estimation for agent-based models for cellular migration and other biological applications is presently active and ongoing. A particular shortcoming of current approaches is the lack of multi-scale summary metrics, despite the fact that ABMs give rise to more multi-scale phenomena than their continuum counterparts. We believe many of these shortcomings can be addressed by the combination of a rigorous functional approach to parameter estimation and the use of topological data analysis summaries of dynamic collective motion data.

4.1.2 Inverse Problems

From the broader perspective of scientific investigation, the process of mathematical modeling is an example of a *forward problem*. This terminology is based on the viewpoint that hypotheses are formed from prior knowledge and then tested with observations. Indeed, this is the common framework for the development of mathematical models; a model is proposed based on prior knowledge and understanding and is subsequently solved, simulated, or otherwise evaluated against data. Any investigation that reverses this order of investigation can thus be thought of as an *inverse problem*. From a mathematical modeling perspective, this means beginning with data and observations and using them to understand the model in question. Albert Tarantola defines inverse problems as a case of "the use of actual observations to infer the properties of a model" in his seminal text on probabilistic inverse approaches to parameter estimation [110].

While the history of inverse problems is rich, and includes ground-breaking advancements in geophysics and tomography in general [112], we are most interested in how the mathematical construction of inverse problem theory allows us to view the task of parameter estimation in a more rigorous light. The common understanding of parameter estimation is as the combination of statistical estimators and optimization applied to the parameters of a mathematical model as the estimands. While this is in no way incorrect, and the tools employed in estimating parameters almost always involve the formal

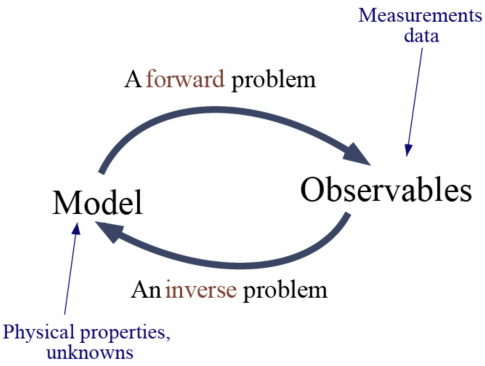

Figure 4.1: A simple diagram illustrating the relationship of inverse problems to forward mathematical modeling problems [111].

treatment of statistics and optimization, it is helpful to understand inverse problems from the perspective of functional analysis and probability theory [113]. By considering a parameter estimate as a solution to an inverse problem, we uncover a more robust framework for understanding the effect of model spaces, noisy random variables, summary statistics, and error models on our parameter estimates.

If the forward problem is stated as $\mathbf{d} = \mathbf{g}(\mathbf{m})$ where $\mathbf{d}$ are the observed data that correspond to the model $\mathbf{m}$, then parameter estimation is—in a particular probabilistic sense—the act of finding a $\mathbf{g}^{-1}$ such that $\mathbf{m} = \mathbf{g}^{-1}(\mathbf{d})$. Inverse problems consider $\mathbf{m}$ and $\mathbf{d}$ in the above relationships to be probability distributions, and through this perspective we may better understand the functional analysis behind the use of common statistical methods such as least squares. This understanding is especially important when applying nonlinear, non-differentiable, and non-invertible operators to data.

While we avoid a rigorous treatment of the inverse problems pertaining to our particular topological operators, we find it important to introduce this framework as it motivates several of our research directions, including the ability to avoid taking derivatives, and the treatment of parameter estimates as samples from a distribution. The aforementioned text by Tarantola [110] supplies ample theory behind the treatment of parameter estimation as an inverse problem, and will be referenced herein.

4.2 The D'Orsogna Model

We focus on the agent-based particle interaction model introduced by M.R. D'Orsogna and colleagues in 2006 [114] as an archetypical example of a collective motion ABM. This seminal model describes the movement of individual particles in swarming systems using

only their locomotion and measures of their attraction and repulsion to other particles in the population. The model leverages fundamental physical laws to characterize properties of motion and interaction, and as a result is often biologically relevant. While this model should certainly not be uniformly applied to all particle interaction studies, it offers a balance of mathematical simplicity and dynamic complexity that makes it ideal for preliminary investigation of ABM techniques.

Thus we use this as our prototype collective motion model, and all of the data and model simulation studies herein are based on slight variants of this model. Because the model takes only position and velocity as input, we need not worry about collecting or calculating other quantities like particle size as part of the data collection process. We also take advantage of the biological interpretability of the model parameters in our subsequent investigation. Finally, we are prompted to choose this model as it has many similarities to the agent-based interaction models used in cell microscopy parameter selection experiments. In particular, it resembles the model that Oleksii Matsiaka and colleagues use to model the migration of cells into scratches, where they are able to estimate parameters of cell diffusivity and cell interactivity strength [15].

4.2.1 D'Orsogna Model Equations and Parameters

The original D'Orsogna model is given by the equations [114]

$$\begin{aligned} \frac{\partial \vec{x}_i}{\partial t} &= \vec{v}_i, \\ m\frac{\partial \vec{v}_i}{\partial t} &= \left(\alpha - \beta|\vec{v}_i|^2\right)\vec{v}_i - \nabla_i U\left(\vec{x}_i\right) \end{aligned} \tag{4.1}$$

where $U(\vec{x}_i)$ is a generalized form of the Morse potential [115] suited for cellular attraction and repulsion:

$$U\left(\vec{x}_i\right) = \sum_{j\neq i} C_r e^{-|\vec{x}_i - \vec{x}_j|/\ell_r} - C_a e^{-|\vec{x}_i - \vec{x}_j|/\ell_a}. \tag{4.2}$$

The N identical particles are indexed by $i \in 1, 2, 3, ..., N$, so in actuality the system consists of $2Nk$ coupled ODEs where k is the dimensionality of the model. We utilize two-dimensional particle motion datasets, and thus consider only the model where $k = 2$, but note that the model generalizes to 1 or 3 dimensions without complication.

To summarize, the dynamics of a particle depend only on its own position $\vec{x}_i$, its own velocity $\vec{v}_i$, and the positions of other particles $\vec{x}_j,\ j \neq i$. The velocity of the particle in each direction is simply the derivative of its position, following the laws of motion. This velocity is affected by two types of forces, the particle's own self-propulsion and the

attractive and repulsive forces placed on the particle by every other particle in the vicinity. α and β are the self-propulsion coefficient and the friction coefficient respectively, which govern how each particle will accelerate to an asymptotic velocity if no other forces are acting on it. All other parameters determine how the particles experience these two-body interactions, with ℓ_a and ℓ_r representing the attractive and repulsive potential ranges respectively, and C_a and C_r providing the magnitude of these forces. This model does therefore assume that all external influences, such as the effect that the environment has on particle motion, are either constant or are entirely captured by the modeled dynamics. For example, the substrate on which the particles are moving would be assumed to have a constant, uniform effect on the particles' locomotion and interactivity, expressed in part by β and C_a respectively.

4.2.2 Nondimensionalization

A subsequent article from the research group behind the original model—shown in equations (4.1) & (4.2)—introduces a nondimensionalized model, which effectively reduces the number of parameters from 7 to 4 and allows the use of unit-less time, position, and velocity variables, yet maintains the same dynamics and structure [116]. By letting $t = \ell_a^2\beta/m\ \bar{t}$, $\mathbf{x}_i = \ell_a\ \bar{\mathbf{x}}_i$, and consequently $\mathbf{v}_i = m/(\ell_a\beta)\ \bar{\mathbf{v}}_i$, we obtain a nondimensionalized model in terms of the dimensionless variables $\bar{t}$, $\bar{\mathbf{x}}_i$, $\bar{\mathbf{v}}_i$ and new parameters $\alpha' = (\alpha\beta\ell_a^2)/m^2$, $m' = m^3/(\beta^2 C_a \ell_a^2)$, $C = C_r/C_a$ and $\ell = \ell_r/\ell_a$:

$$\begin{aligned} \frac{\partial \bar{\mathbf{x}}_i}{\partial \bar{t}} &= \bar{\mathbf{v}}_i, \\ \frac{\partial \bar{\mathbf{v}}_i}{\partial \bar{t}} &= \left(\alpha' - |\bar{\mathbf{v}}_i|^2\right)\bar{\mathbf{v}}_i - \frac{1}{m}\nabla_i U\left(\bar{\mathbf{x}}_i\right), \\ U\left(\bar{\mathbf{x}}_i\right) &= \sum_{j\neq i} C e^{-|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j|/\ell} - e^{-|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j|}. \end{aligned} \tag{4.3}$$

This more reasonable number of parameters is our primary motivation for employing the nondimensionalized model. With only 4 parameters, we may either fix or estimate each parameter, rather than having to ignore parameters that do not contribute to the analysis. Furthermore, these new parameters are more interpretable from a biological standpoint, and allow for simpler parameter space assumptions. For example, several sources, including the aforementioned follow-up paper, note that only long-range attraction and short-range repulsion are biologically relevant [116]. By this we know that we must consider $C \geq 1$ and $\ell \leq 1$, and we thus utilize these constraints when choosing parameter values for simulation and intervals for parameter estimation. While we do not intend for

this specific nondimensionalized D'Orsogna model to be fitted to cell migration data, we choose it because it exhibits many of the same dynamics as models which are suited for the purpose. Finally, we will exclude any extraneous notation in our candidate models, as we will not be directly comparing dimensional and dimensionless models.

4.3 Related Work

Our primary objective is to expand upon the ideas presented in an article by D. Bhaskar and colleagues. This paper demonstrates the efficacy of TDA for the categorization of agent-based collective motion simulations into individual *phenotypes*, or long-time emergent behavior patterns [117]. This work leverages the aforementioned crocker of Topaz [97], which summarizes the persistent homology of a time series of point clouds, to categorize the distinct phenotypes of the D'Orsogna model. We thus summarize the results and findings of both this investigation and other relevant TDA research.

4.3.1 Topology and Emergent Behaviors

While the research applying crockers to aphid models is based on ABMs [98], the data in question are largely devoid of the emergent characteristics that make ABMs uniquely challenging due to the relatively small number of agents and the weak interactive forces. In contrast, we aim to apply topological data analysis to the task of ABM parameter estimation because TDA best describes the emergent behaviors which are intrinsic to agent-based models while still being sensitive to local interactions and small-scale dynamics. This ability to describe multi-scale properties is particularly important for time-varying data such as those described by crockers, since these properties typically emerge over sufficiently long time spans.

TDA's sensitivity to agent-level behavior is immediately apparent. The construction of simplicial complexes is based on pairwise connections between sufficiently-near agents, and while the impact of these individual connections is effectively "smoothed" when counting higher-dimensional holes rather than individual connections, these building blocks still form the basis of all TDA. The ability of TDA to detect emergent behaviors is based substantially on the persistence of topological features. These features that persist over large proximities are often tied to large-scale properties, such as clustering [118] and uniform velocity [117].

Recent research demonstrates more profound connections between persistent homology and emergent behaviors by formalizing dynamic data in a topologically consistent

manner via *dynamic metric spaces* [119]. This lead to the development of specific TDA tools for the analysis of dynamic flocking data [120], further showing the usefulness of combining persistent homology and time-varying data. Altogether, the efficacy of TDA for encapsulating the emergence present in agent-based models makes it an ideal tool for summarizing many-agent model output.

4.3.2 Crockers and Parameter Recovery

A recent paper by Dhananjay Bhaskar takes the first step toward bridging the gap between speculative investigation of dynamic point cloud data using crocker plots and a full parameter estimation pipeline using vectorized crockers (crocker matrices) [117]. Its primary contributions are the novel application of TDA in the form of crocker matrices to the task of simulated D'Orsogna model phenotype classification, and showing that this approach outperforms the use of purpose-built order parameters for the same task.

While the original D'Orsogna model paper [114] describes the different emergent behaviors of the model according to the location of the parameters in their phase diagram, this paper further separates these behaviors into *phenotypes*, a term referring to the "paradigmatic collective motion behaviors" of the simulated particles [117]. The aforementioned follow-up article [116] does use more descriptive terminology in naming these distinct phase diagram regions, but to avoid confusion we utilize the clearly-demarcated names from the Bhaskar work. These phenotype names largely correspond to self-explanatory perceptible group actions, including "mills", "rings", "swarms", and "escapes". The study specifically examines 25 combinations of model parameters, each of which fit into one of the 6 model phenotypes.

Order parameters are measures of specific intra-population behaviors often used in collective motion studies. This study uses 4 such order parameters—polarity, angular momentum, absolute angular momentum, and mean nearest neighbor distance—all of which are used in the D'Orsogna continuation article to examine phase transition events. By comparing the performance of these order parameters which are chosen specifically for this model to the performance of crocker matrices they are able to highlight one of the most significant features of TDA: that it requires no *a priori* knowledge of the data and no model-specific tuning.

To directly compare the performance of these order parameters to the performance of the crockers, the study uses both as vectorized input into two parameter recovery algorithms (Figure 4.2). The first approach is unsupervised k-medoids clustering of the inputs with $k = 25$ corresponding to the number of parameter categories. The

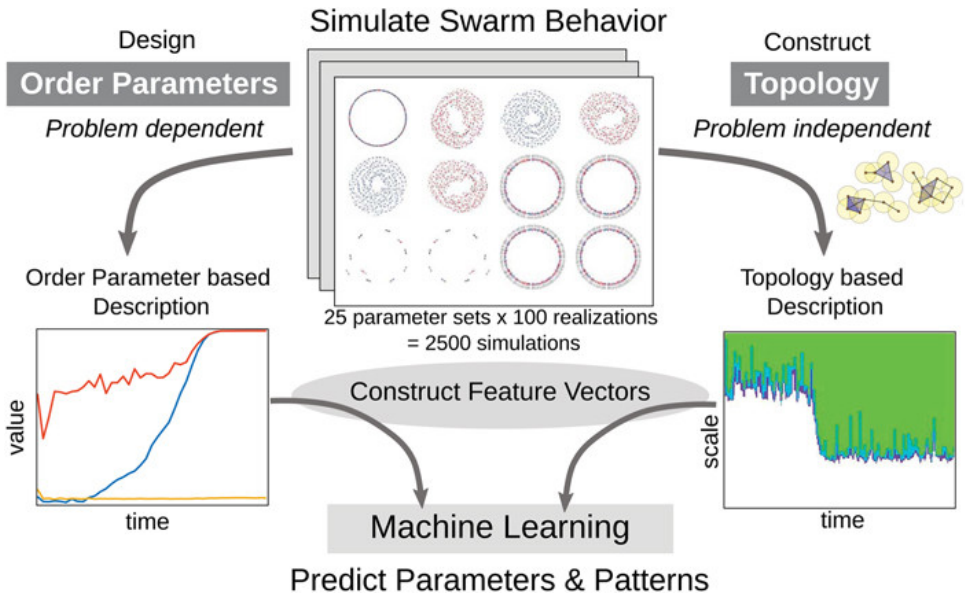

Figure 4.2: The experimental procedure of the Bhaskar et al. study [117].

second approach is a multi-class linear support vector machine (SVM) classification—a supervised machine learning algorithm. The final outcome of the study was the result that using crockers as input into either algorithm resulted in better classification results for both phenotype categories and parameter categories, even when accounting for the higher dimensional input data using principle component analysis. Overall, this study is instrumental for proving the worth of crocker-based TDA for the classification of agent-based particle motion data.

4.3.3 Inverse Problems with TDA

The primary difference between our goal of parameter estimation, and the previously-outlined approaches that leverage TDA is the utilization of the formal mathematical framework of inverse problems. While the above research demonstrates that TDA is useful for clustering and classification tasks, both of which are based on the optimization of discrete statistical measures of comparison, we are most interested in the capability of TDA operators to work within a framework of continuous parameter estimation. While the in-depth study of the utility of topological operators for inverse problems is especially young, there are a number of promising preliminary results, particularly in regards to persistent homology.

A recent survey paper by Steve Oudot and Elchanan Solomon summarizes the steps taken toward building topological operators that have defined inverses [121]. With a

focus on interpretability and explainability, they introduce the investigations into the bijectivity of persistence modules with respect to the point clouds that generate them. By providing a method for taking the derivative of appearance times of edges of homological features with respect to the points that bound them, they show that point clouds can be iteratively recovered from their persistence modules via a Newton-method approach.

While we do not utilize of any of these methods, instead choosing to forego the use of derivative-based optimization in favor of direct-search algorithms, these results show the unique capabilities of TDA in the space of inverse problems. The formal treatment of persistence inverses requires significantly more investigation, but we hope that, by using persistent homology to estimate parameters, we are able to contribute another piece of evidence toward the argument for the continued application of TDA to inverse problems.

4.4 Methods

We herein explore how TDA may be applied to the task of parameter estimation for agent-based-models. In particular, we investigate how we may extend the persistent homology techniques that are used in the Bhaskar paper's discrete categorization task into the more formal realm of inverse problems and the continuous fitting of ABM parameters to data. We leverage the probabilistic underpinning of least-squares regression to allow for the systematic treatment of parameters and stochastic noise as random variables, and frame the task of parameter estimation as an inverse problem. To this end, we apply TDA functions that produce Betti numbers to simulations of D'Orsogna-type ABMs (4.1), including a stochastic formulation of the model which introduces biologically-realistic noise to the particle trajectories, and investigate methods for recovering model parameters by minimizing topological error.

4.4.1 Candidate Models

We utilize individual-particle-based models, as we believe their ability to capture both small-scale interactions and emergent behaviors is paramount to accurately modeling a range of population dynamics. The D'Orsonga model in particular is able to model particle adhesion, persistent locomotion, and uniform swarming. The emergent behaviors this model produces are primarily captured by the aforementioned model phenotypes. While a phenotype change occurs due to larger-scale parameter changes than we expect to encounter, it is still important that our TDA can recognize these emergent behaviors. The D'Orsogna model is simplistic in its construction compared to other contemporary

ABMs like the ones used in economics [122] and sociology [123]. We believe, however, that it is an ideal prototype for investigating the efficacy of TDA for the estimation of ABM parameters due to its simplicity, and will herein introduce slight modifications to address its practical shortcomings.

Stochastic D'Orsogna Model

The D'Orsogna model has the limitation of being a *deterministic model*, meaning that the same inputs at the first time step will produce the same outputs at every successive time step. This property is desirable at some levels of investigation because it results in consistent large-scale dynamics for similar inputs and is easier to simulate and examine analytically. Biological systems, however, are almost always better modeled by *stochastic models*, which incorporate randomness into their outputs. This is epitomized by the Brownian motion that is observed in all varieties of many-particle systems (see 2.5). Since mathematical models are never able to capture every cause that contributes to the behavior of agents in a system, it is often better to attribute some of the behaviors to detectably random noise.

We therefore introduce a stochastic variant of the D'Orsogna model by allowing the particles to take random walks. The formulation is identical to the analysis used in Section 2.5. To recap, we introduce random walks using the Itô calculus [77], which is easier to mathematically formalize compared to the Langevin method of simple additive noise. This involves formulating the D'Orsogna model as a system of stochastic differential equations (SDEs) rather than as ordinary differential equations. The self-propulsion and interaction terms are allowed to remain fully deterministic as they are given in the original model.

The stochastic term is added to the definition of velocity to allow for random perturbations to a particle's position, which then propagates through the deterministic force equations. As with all SDEs in the realm of Itô calculus, we do this by leaving the deterministic terms with their deterministic differential ($\mathrm{d}t$) and introduce the stochasticity via the random variable differential in the stochastic term. We use the nondimensionalized model (4.3) to reduce the number of parameters, dropping the extraneous notation as previously noted. Thus our stochastic D'Orsogna particle interaction model is:

$$\begin{aligned} \mathrm{d}\vec{X}_i &= \vec{v}_i \, \mathrm{d}t + \vec{\sigma} \, \mathrm{d}W \\ \frac{\partial \vec{v}_i}{\partial t} &= \left(\alpha - |\vec{v}_i|^2\right) \vec{v}_i - \frac{1}{m} \nabla_i U\left(\vec{x}_i\right), \\ U\left(\vec{x}_i\right) &= \sum_{j \neq i} C e^{-|\vec{x}_i - \vec{x}_j|/\ell} - e^{-|\vec{x}_i - \vec{x}_j|}. \end{aligned} \tag{4.4}$$

with $\vec{x}_i$ being a realization of the random variable obtained from $\vec{X}_i$ at a given time. Because we are considering a 2-dimensional system, $\vec{x}_i, \vec{v}_i \in \mathbb{R}^2$. Further, $\vec{\sigma} = [\sigma, \sigma]$ where σ is a random walk diffusion coefficient. The system of SDEs involves only a single Wiener process $W \sim \mathcal{N}(0, t)$. The resultant stochastic processes $\vec{X}_i$ thus consist of two-dimensional random variables for each particle indexed by i with each dimension corresponding to the position of the particle in each of the two Cartesian directions. This model incorporates both the particle's deterministic interactions—as in the standard D'Orsogna model—and the random position perturbations the particle experiences in both spatial directions.

Candidate Model Parameters

We highlight the fact that the corresponding dimensional form of the equation can be recovered by analyzing the dimensions of σ in a manner similar to how the nondimensional equivalent to velocity is produced. Returning to the notation of (4.3), we say that $\bar{z}$ is the nondimensional version of the parameter or variable z. Just as we may recover the units of velocity by substituting the characteristic expressions for $\bar{x}$ and $\bar{t}$ into the equation $\partial\bar{x}/\partial\bar{t} = \bar{v}$, we can find the characteristic unit for σ by solving $\mathrm{d}\bar{x}/\mathrm{d}\bar{W} = \bar{\sigma}$. Note that, since $\bar{W} \sim \mathcal{N}(0, \bar{t})$ in this notation, we have $[W] = \left[\sqrt{t}\right] = \left[\ell_a\sqrt{\beta}/\sqrt{m}\ \sqrt{\bar{t}}\right]$. Therefore

$$\bar{\sigma} = \frac{\mathrm{d}\bar{x}}{\mathrm{d}\bar{W}} \tag{4.5}$$

$$= \frac{1}{\ell_a}\frac{\ell_a\sqrt{\beta}}{\sqrt{m}}\frac{\mathrm{d}x}{\mathrm{d}W} \tag{4.6}$$

$$= \frac{\sqrt{\beta}}{\sqrt{m}}\sigma. \tag{4.7}$$

Thus $\sigma = \sqrt{m}/\sqrt{\beta}\ \bar{\sigma}$, which results in the correct units for dimensional σ of $\mathrm{kg}/\sqrt{\mathrm{s}}$ assuming SI units throughout. This dimensional form of σ corresponds to the random walk parameter in Section 2.5, and can be used for these types of analyses.

Next we fix $m = 1$ in order to remove any explicit references to mass in the equations. This does not have the effect of simply asserting the unit mass of the particles as it would for the original model (4.1). Instead we are asserting the expression $m = m'^3/(\beta^2 C_a \ell_a^2) = 1$, which implies that $m'^3 = \beta^2 C_a \ell_a^2$ where m' is the mass of the particles (denoted by m in the original model). This is a perfectly reasonable assumption to make from a physical perspective, as particles with more mass will most certainly experience greater friction forces (β) and produce stronger attraction (C_a and ℓ_a). The equation simply formalizes the proportions in which these qualities interact. This choice is further justified by previous

work, which fixes m under the interpretation of the parameter as affecting "the time scale of the particle interaction" [116]. By fixing m we are able to focus on choosing ℓ and C in so that the particles exhibit the desired interaction dynamics without simultaneously tuning m.

Boundary Conditions

Finally, we would like to prevent the particles from simply *escaping* from the frame. This escape behavior is a likely outcome for realistic model parameters according to D'Orsogna model phenotype investigation [117]. To accomplish this we impose a zero-flux condition on the particle population. This condition is most applicable for systems with physical boundaries on all sides, such as a group of people in a room with a closed door, or a cell monolayer contained in a vessel. While we use a square domain and therefore a square boundary, the same principles can be applied to any feasible spatial domain. However this is also reasonable for systems with no boundaries but which have no flux due to an equal number of agents entering and leaving the population. This is a common assumption for many biological models, similar to the assumption of an equal birth and death rate in basic SIR models [124]. For this particle motion model it is better to think of a particle as being "replaced" by a new particle moving into frame at an equal speed in the opposite direction rather than assuming the boundary condition causes particles to "bounce off" the boundary.

Mathematically, we impose this condition by enforcing a particular sign for the relevant component of a particle's velocity if its position is at or beyond the boundary. For example, if the particle crosses the lower boundary in the x_1 direction, we force the first component of its velocity to be positive leaving the second component unchanged, effectively pointing the particle back toward the domain. This results in no particles leaving or entering through any of the boundaries. We acknowledge that imposing the condition in this way has shortcomings. In particular the persistent locomotion allowed by the model is limited to the dimensions of the domain. However, this method prevents the discontinuous intercellular interactions that would arise from any type of periodic boundary conditions that produce a flux of zero.

Candidate Model Equations

Our set of assumptions results in two ABMs that we use for data simulation and parameter estimation experiments. The deterministic model (ODE system) is

$$\begin{aligned}
\frac{\partial \vec{x}_i}{\partial t} &= \begin{cases} \vec{v}_i & \vec{x}_i \in \Omega \\ -\operatorname{sgn}(\vec{v}_i)_1 \ \hat{i} \cdot \vec{v}_i & (\vec{x}_i)_1 > x_b \\ \operatorname{sgn}(\vec{v}_i)_1 \ \hat{i} \cdot \vec{v}_i & (\vec{x}_i)_1 < x_a \\ -\operatorname{sgn}(\vec{v}_i)_2 \ \hat{j} \cdot \vec{v}_i & (\vec{x}_i)_2 > y_b \\ \operatorname{sgn}(\vec{v}_i)_2 \ \hat{j} \cdot \vec{v}_i & (\vec{x}_i)_2 < y_a \end{cases} \\
\frac{\partial \vec{v}_i}{\partial t} &= \left(\alpha - |\vec{v}_i|^2\right) \vec{v}_i - \nabla_i U\left(\vec{x}_i\right) \\
U\left(\vec{x}_i\right) &= \sum_{j \neq i} C e^{-|\vec{x}_i - \vec{x}_j|/\ell} - e^{-|\vec{x}_i - \vec{x}_j|}
\end{aligned} \tag{4.8}$$

where the N identical particles are indexed by $i \in 1, 2, 3, ..., N$ and $\Omega = [x_a, x_b] \times [y_a, y_b]$ is the positional domain.

The stochastic model (SDE system) is

$$\begin{aligned}
\mathrm{d}\vec{X}_i &= \vec{\sigma} \ \mathrm{d}W + \begin{cases} \vec{v}_i \ \mathrm{d}t & \vec{x}_i \in \Omega \\ -\operatorname{sgn}(\vec{v}_i)_1 \ \hat{i} \cdot \vec{v}_i \ \mathrm{d}t & (\vec{x}_i)_1 > x_b \\ \operatorname{sgn}(\vec{v}_i)_1 \ \hat{i} \cdot \vec{v}_i \ \mathrm{d}t & (\vec{x}_i)_1 < x_a \\ -\operatorname{sgn}(\vec{v}_i)_2 \ \hat{j} \cdot \vec{v}_i \ \mathrm{d}t & (\vec{x}_i)_2 > y_b \\ \operatorname{sgn}(\vec{v}_i)_2 \ \hat{j} \cdot \vec{v}_i \ \mathrm{d}t & (\vec{x}_i)_2 < y_a \end{cases} \\
\frac{\partial \vec{v}_i}{\partial t} &= \left(\alpha - |\vec{v}_i|^2\right) \vec{v}_i - \nabla_i U\left(\vec{x}_i\right) \\
U\left(\vec{x}_i\right) &= \sum_{j \neq i} C e^{-|\vec{x}_i - \vec{x}_j|/\ell} - e^{-|\vec{x}_i - \vec{x}_j|}
\end{aligned} \tag{4.9}$$

where the N identical particles are indexed by $i \in 1, 2, 3, ..., N$, $\vec{x}_i(t) \sim \vec{X}_i(t)$, and $\Omega = [x_a, x_b] \times [y_a, y_b]$ is the positional domain.

These models simulate populations of particles that are self-propelled, interact with each other via two-body attraction and repulsion, take random walks, and do not leave the population due to zero-flux boundaries. While there are almost certainly other forces acting on particles in any real population of biological particles, we are confident this model exhibits sufficiently complex dynamics to act as a prototype of emergent ABMs for the sake of TDA-based parameter estimation.

4.4.2 Simulation Details

Datasets are created by approximately solving the models using appropriate numerical methods. We use the 4th-order Runge-Kutta method [125] for the ODE system (4.8) and the Euler-Maruyama method [126] for the SDE system (4.9). In particular, we use the adaptive step size Dormand-Prince 5th-order correction method [127] as provided by the SciPy package for Python [128]. The Euler-Maruyama method is the extension of the simple Forward Euler discretization scheme to stochastic processes. Given a general SDE

$$\mathrm{d}X_t = a(X_t, t)\,\mathrm{d}t + b(X_t, t)\,\mathrm{d}W_t, \tag{4.10}$$

with initial condition $X_0 = x_0$ we approximate the solution using the Euler-Maruyama method over a discretization $0 = t_0 < t_1 < ... < t_N = T < \infty$ of the interval $[0, T]$ by defining the Markov chain Y_n:

$$\begin{aligned} Y_0 &= X_0, \\ Y_{n+1} &= Y_n + a(Y_n, t_n)(t_{n+1} - t_n) + b(Y_n, t_n)(W_{t_{n+1}} - W_{t_n}) \end{aligned} \tag{4.11}$$

where $W_{t_{n+1}} - W_{t_n} \sim \mathcal{N}(0, t_{n+1} - t_n)$ [126]. Thus we randomly sample the stochastic differential at each time step. We utilize a fixed time step such that $t_{n+1} - t_n = \Delta t/100\ \forall n = 0, ..., N-1$ where Δt is the fixed time step of the dataset.

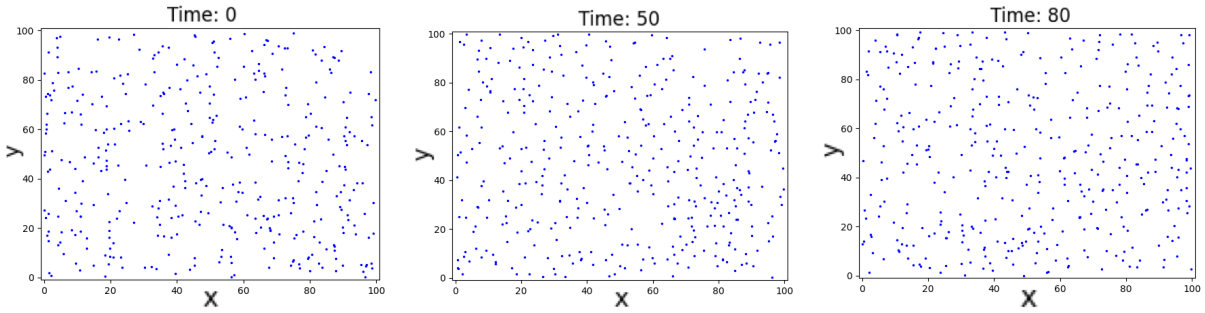

Figure 4.3: Plots of particle positions from a stochastic model simulated dataset. Initial frame (left), middle frame (center), and late frame (right) shown.

We set the spatial domain as $\Omega = [0, 100]^2$ and utilize the aforementioned boundary conditions. Each dataset consists of a population 400 particles, randomly initialized such that $\vec{x} \sim \mathcal{U}([0, 100]^2)$ and $\vec{v} \sim \mathcal{N}(\vec{0}, 0.2\mathbf{I})$. Initial datasets are integrated over $t \in [0, 100]$ with $\Delta t = 1$ such that there are 100 simulated frames. We subsequently analyze the time

scale of both deterministic and stochastic simulations and reduce the time span of the datasets we investigate for the purposes of parameter estimation. In short, we find that the particles are sufficiently "mixed" toward the end of the original $[0, 100]$ time span and thus we focus our analyses on shortened datasets. We collect the positions and velocities of each particle at each frame, resulting in up to 101 separate clouds of 400 points in 4-dimensional space. Because these point clouds are collected discretely in time, in most cases they are treated as individual, distinct observations (see Figure 4.3).

Table 4.1: Fixed values of the deterministic parameters used in the model simulations.

Par.	**Val.**	**Significance**
α	0.3	relative strength of self-propulsion with respect to interactive forces
C	1.8	ratio of repulsive strength to attractive strength
ℓ	0.4	ratio of repulsive radius to attractive radius

We choose values for the three deterministic model parameters based on the "biological realism" constraint of $C > 1,\ \ell < 1$ [116] and on achieving the desired visible dynamics of particle adhesion, persistent individual locomotion through the population, and a lack of unwanted steady-state global behavior such as clumping (Figure 4.4) or uniform milling. The process of selecting parameters to match certain behaviors is perfectly reasonable due in part to the similar order of magnitude between parameters as a result of nondimensionalization. Furthermore, an understanding of the effect each parameter has on the behavior of the system ([116], see Table 4.1) allows for straightforward tuning.

Time Span Analysis

As the focus of our research is to recover the model parameters of our simulated datasets, we need to ensure that the time span of our experiments is appropriate for the task. In particular, we want to make sure there is a significant relationship between the model outputs of all collected time points and the model parameters. This also has implications for the collection of real data, as most researchers would like to avoid the expenses of collecting unnecessary experimental data. Thus, we examine the correlation between small deviations in the model parameters and the resulting change in the quantities that the model produces, namely the positions and velocities of the particles.

Using simulated data affords us the ability to compare the position $\hat{\mathbf{x}}$ and velocity $\hat{\mathbf{v}}$ of a particle in a *perturbed* dataset to the position $\mathbf{x}$ and velocity $\mathbf{v}$ of the same particle in

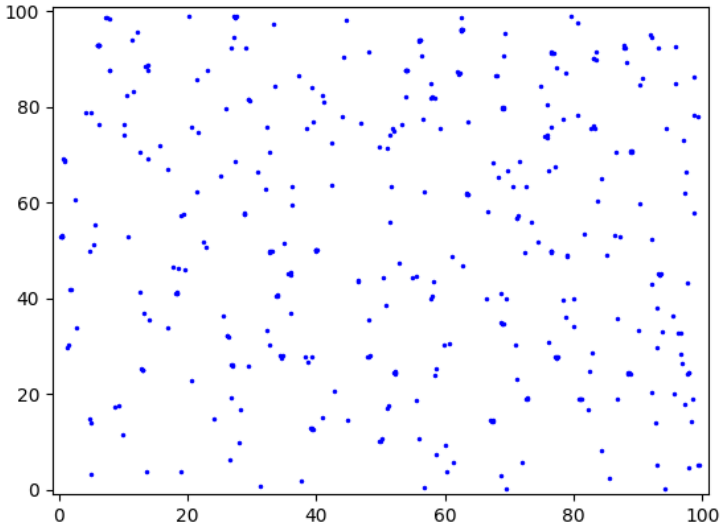

Figure 4.4: A plot of particle positions from a simulated dataset whose parameters cause distinct clumping behavior.

a *ground truth* dataset at any time t without any additional data processing or labeling. We could collect these comparisons for each particle individually to obtain 400 separate measures of model output change over time, but we are most interested in the summary behavior of the system at each time step. Thus we examine measures which take the square root of the average over all particles of the squared differences between particle properties at a given time. This is of course directly related to the idea of *root mean squared error*, and we utilize this summary due to its ability to penalize outliers while remaining approximately linear in scale with respect to the individual measurement. For the positions we simply take the root of the mean of the squared differences summed over the components:

$$P(t) = \left(\frac{1}{400} \sum_{i=1}^{400} \| \left(\mathbf{x}_i(t) - \hat{\mathbf{x}}_i(t) \right)^2 \|_1 \right)^{1/2}. \tag{4.12}$$

For the velocities we take the root of the mean of the squared 2-norm of the vector difference:

$$V(t) = \left(\frac{1}{400} \sum_{i=1}^{400} \|\mathbf{v}_i(t) - \hat{\mathbf{v}}_i(t)\|_2^2 \right)^{1/2}. \tag{4.13}$$

These may be accurately called "metrics" on the space of position and velocity point clouds respectively, as they yield a non-negative measure of the distance between two point clouds.

Our initial investigation involves changing the model parameters one at a time by $\pm 1\%$,

$\pm 5\%$, and $\pm 10\%$, simulating the corresponding ODE dataset over the aforementioned time span, and computing the above functions pairwise for these perturbed datasets with the original ground-truth dataset. In this manner we are collecting time series $P(t;\theta)$ and $V(t;\theta)$, one for each value of the 6 values of the 3 deterministic model parameters, resulting in 18 total time series. We first plot these function values per frame with each plot arranged by parameter value with a value of zero in the center corresponding to the unchanged parameter, hoping to see a clear correlation between an increase in changes to the parameters and larger metric values. We refer to these plots colloquially as *trench plots*, as a trench or "V" shape to the bars equates to a strong and consistent correlation between parameter value and metric, and is thus the ideal visual shape of the plot. A subset of the trench plots created for this investigation for the interactive strength parameter C is shown in Figure 4.5. The trench plots for both α and ℓ using the same subset of frames are included in Appendix B.1.

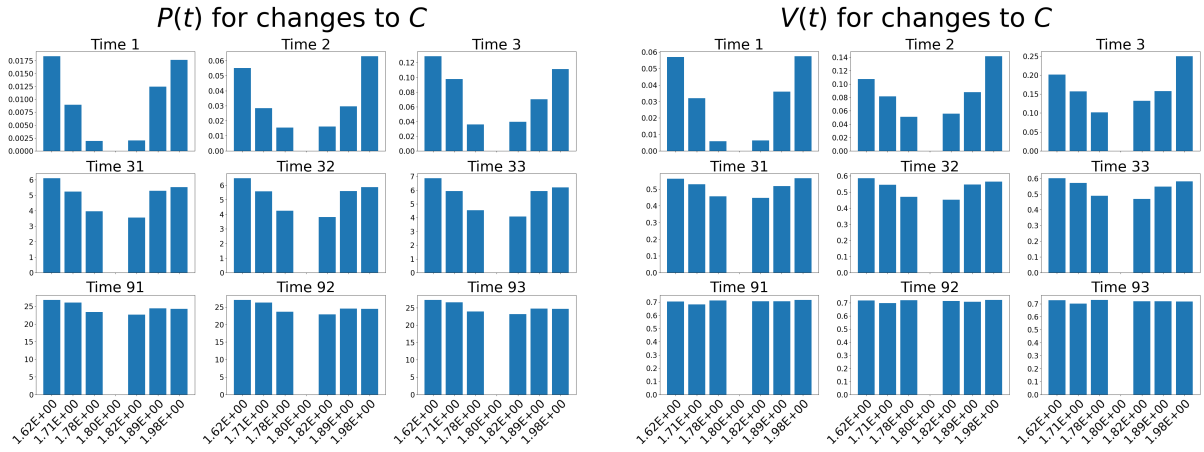

Figure 4.5: A selection of relevant trench plots for the position metric (left) and velocity metric (right) for parameter C. The earliest frames (first column) show the strongest correlation, slightly later frames (second column) show weaker yet still significant correlation, and the latest frames (third column) demonstrate a distinct lack of correlation.

While these plots do convincingly illustrate a strong correlation between changes in model parameter and changes in metric, we still need a method of quantifying this correlation. This process of measuring the relationship between model inputs and model outputs—or functions thereof—falls under the broadest definitions of *sensitivity analysis* [129]. However, we refrain from using this terminology as most formal sensitivity analyses involve the derivatives of models with respect to the parameters, a step that we avoid

taking due to the limitations of TDA. Instead we will attempt to quantify this apparently linear relationship using the *Pearson correlation coefficient* as provided by SciPy [130]. This coefficient, often denoted r, is simply the covariance between two variables normalized to $[-1, +1]$ in such a way that it measures the "strength" of the linear relationship between them, and has in fact been used as a measure of parameter sensitivity for decades [131]. For our cases we can summarize the correlation between the absolute value of the change in the model parameter and the aforementioned metrics using the correlation coefficient, with values closer to 1 demonstrating a stronger correlation. Therefore we compute the r for each frame using the 7 pairs of values as shown in each trench plot, and plot the values of r over time to best understand how this relationship degrades as the data continue to evolve. The plots of r for the parameters C and α, obtained using the position and velocity metrics, are shown in Figure 4.6.

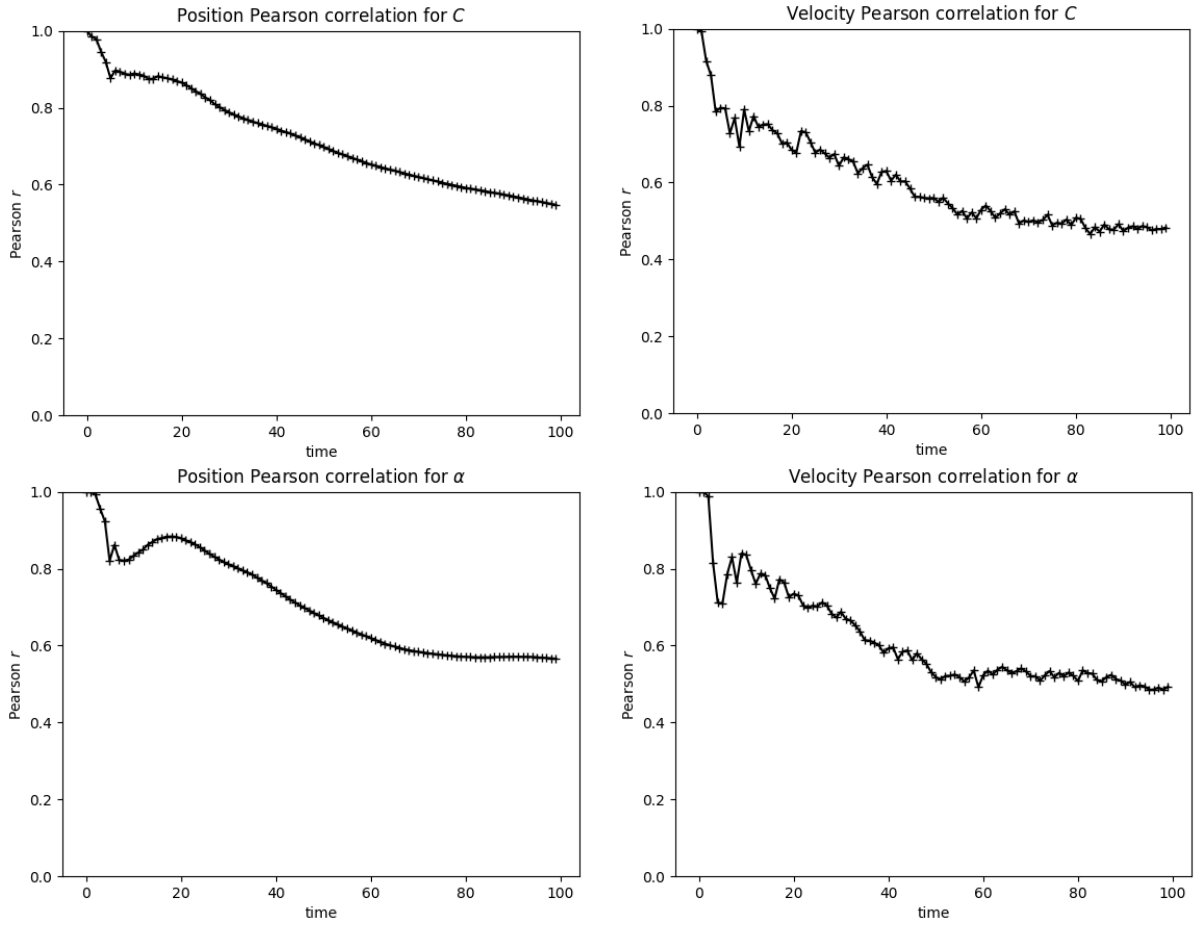

Figure 4.6: Plots of the Pearson correlation coefficients for each time step of the position (left) and velocity (right) trench plots for the model parameters C (top) and α (bottom).

As expected, the correlation does decrease over time for all parameters with respect to either metric. Distinct features of these plots include the sharp decrease over the first 4 or 5 time steps and a steady but significantly sub-linear decrease over the remaining time. We also note that, while the values of r for the velocity metric are both noisier and comparably lower than the values for the position metric, the overall pattern is clearly the same. Equipped with these results, we may draw some conclusions about the importance of individual time steps for summarizing the effect that changing a model parameter has on the outputs of the system. We certainly want to include at least the first 5 time steps in our analyses, though having only 5 observations over time is likely insufficient. Based upon the qualitative "rebound" behavior in the correlation plots for both metrics and all three parameters (Figures 4.6 and B.3), we choose to use all time steps up to and including $t = 20$, which equates to integrating over the interval $t \in [0, 20]$ using the appropriate numerical method.

Random Walk Parameter σ

The process of choosing a sensible value for the diffusive random walk parameter σ, which directly controls the magnitude of the noise, requires further investigation. In particular, we need a level of noise that is substantial enough to be biologically realistic, but also does not overwhelm the other modeled behaviors of self-propulsion and particle interaction. We determine this balance using the same investigation employed in determining the time span, but instead of using the ODE system to simulate datasets we use the SDE system and examine the effect that increasing σ has on the trench plots and average Pearson correlation of the metrics over our now-fixed time span. Thus we will select a value for σ that does not entirely obscure the relationship between model parameters and our above metrics, asserting that this reasonable level of noise will allow for a fair evaluation of our subsequent parameter estimation methods. Note that for all datasets $\vec{\sigma} = [\sigma, \sigma]^T$, as the random walks are assumed to be equal in both x and y directions.

Thus we calculate the value of r for each parameter at each time step as detailed above for varying levels of σ. We vary the noise parameter σ by performing a manual grid search. Starting at $\sigma = 10^{-3}$ we inspect the effect that the noise has on the Pearson r plots for each parameter, and then increase the noise by an order of magnitude if the effect of the noise is insufficient for a reasonable error study. As previously mentioned, this method of examining correlation is a suitable method for ascertaining the sensitivity of the model parameters with respect to the chosen metric. While we will continue to avoid a thorough study of parameter sensitivity and thus refrain from making statements about the relative sensitivities of our 3 model parameters, we do note that the Pearson

plot for the self-propulsion parameter α is more significantly impacted by the addition of noise with respect to both aforementioned metrics, and thus use it as the primary reference point for choosing a value for σ.

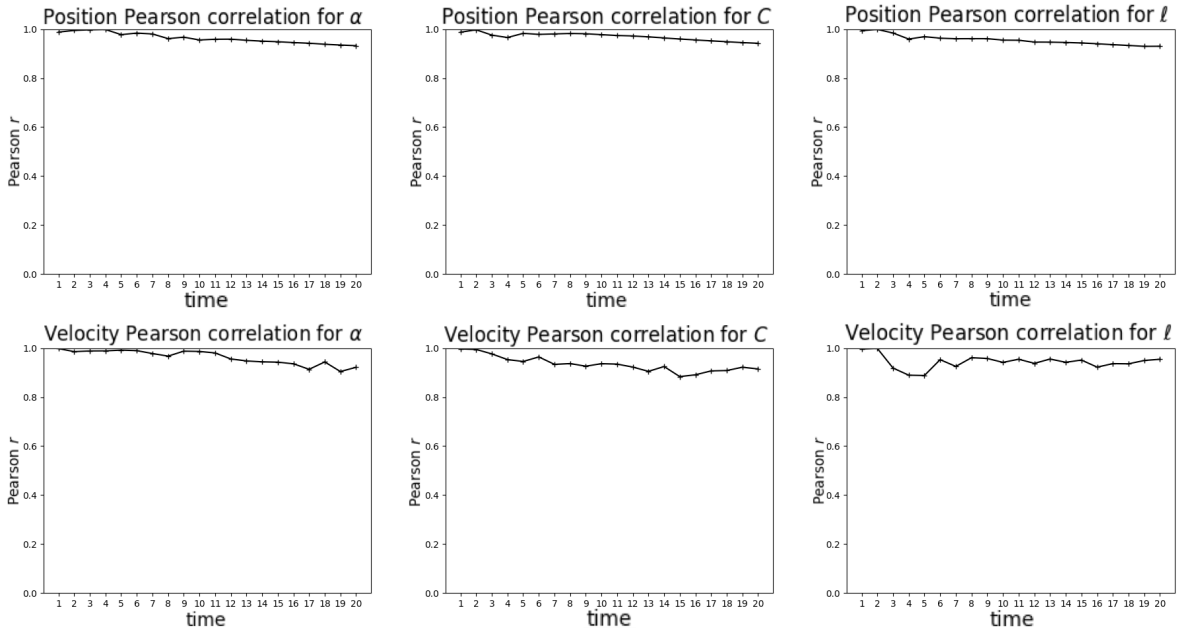

Figure 4.7: Pearson correlation plots for datasets simulated with $\sigma = 10^{-3}$ showing a near-perfect correlation between changes to the model parameters and either metric.

The correlation for $\sigma = 10^{-3}$ is nearly perfect for both metrics and all three parameters (Figure 4.7). In fact, the Pearson r for this low-noise investigation is almost always higher than the r values for the noiseless model. Upon inspecting the corresponding trench plots (see Figures 4.8, B.4, B.5), we conclude that this behavior is likely due to the non-zero metric values for the true value of the parameter. While this does mean that there is never an exact match for the model parameter from an estimation standpoint, this datapoint is a better fit from a linear correlation perspective than the persistent zero value that the ODE system produces.

This behavior does certainly suggest that this method of investigation has limitations in terms of effectiveness on different types of data, but in any case confirms that there is too strong a correlation for this data to be considered noisy. Thus we increase σ by a factor of 10 to $\sigma = 10^{-2}$. This level of noise causes a significant reduction in correlation for the most-sensitive α model parameter. It also introduces a surprising behavior of causing the α correlation to increase over time. This is likely to be due to the explicit

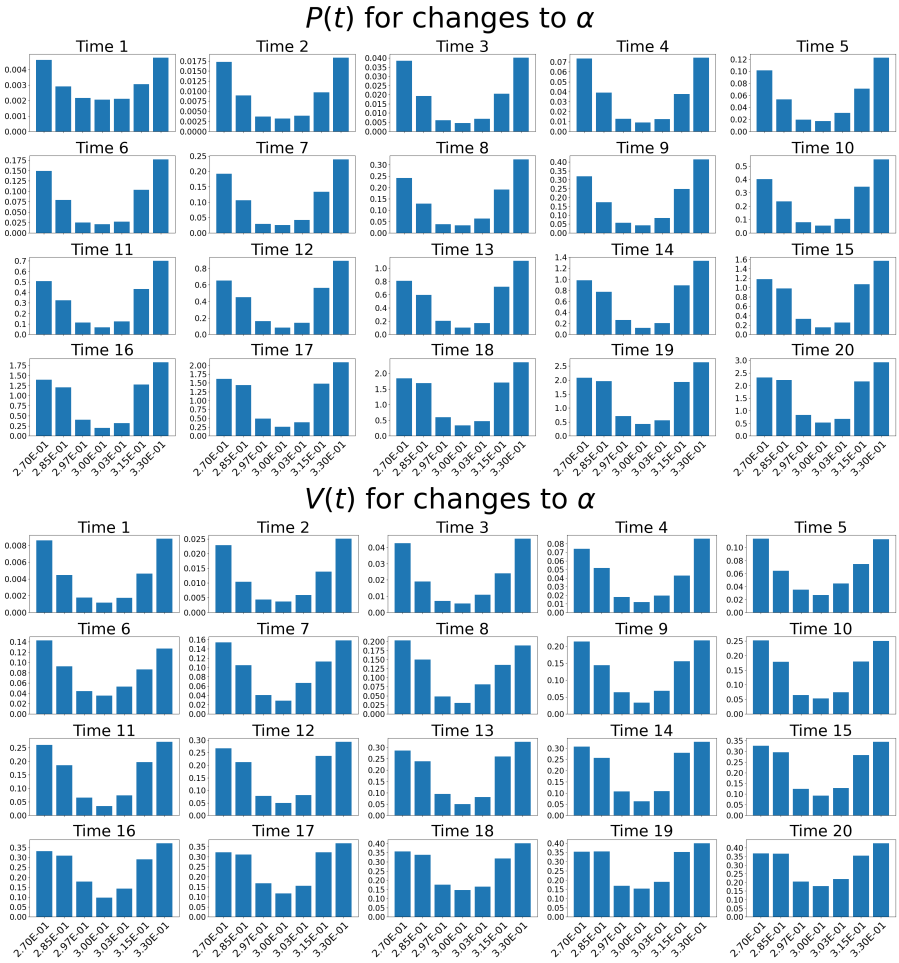

Figure 4.8: Trench plots for $\sigma = 10^{-3}$ illustrating the clear correlation for the α model parameter (Figure 4.7, column 1).

link between α and self-propulsion persistence. In any case, the correlation plots for the other model parameters are still exceedingly strong for the other parameters as shown in Figures 4.9 and B.6, so we increase σ once more and use $\sigma = 10^{-1}$. At this level of noise the correlation is almost completely eliminated for all parameters. Because the value of r becomes negative several times over the time span, the lack of correlation is best illustrated in the trench plots, which show that the noise completely dominates the signal (Figures 4.10 and B.7).

Based on these results, we choose to split the difference between these noise levels and set $\sigma = 0.05$. Using the same analysis employed in Section 2.5, we can say that this corresponds to particles randomly diffusing according to $\mathcal{N}(x_0, \sigma^2 t)$ in either direction where x_0 is the particle's initial position in that dimension and t is the time with the initial condition at $t = 0$. Knowing this distribution, we can conceptualize how far a particle is

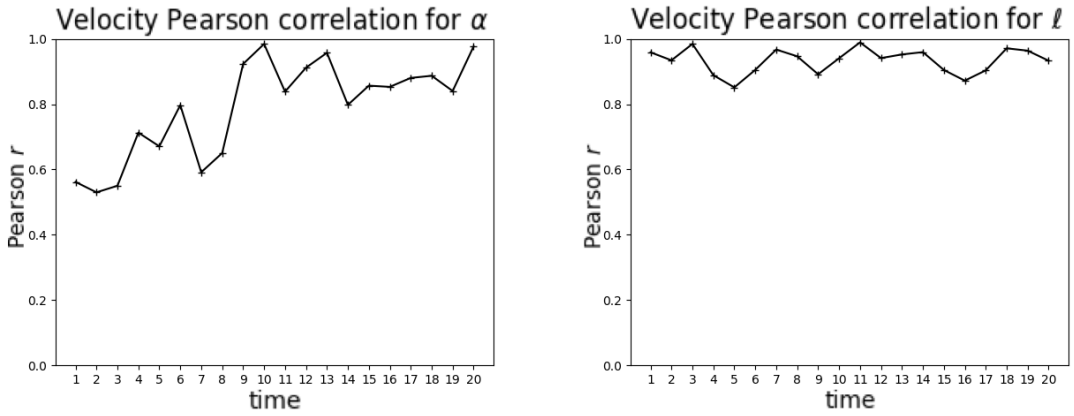

Figure 4.9: Correlation plots for α (left) and ℓ (right) from data where $\sigma = 10^{-2}$.

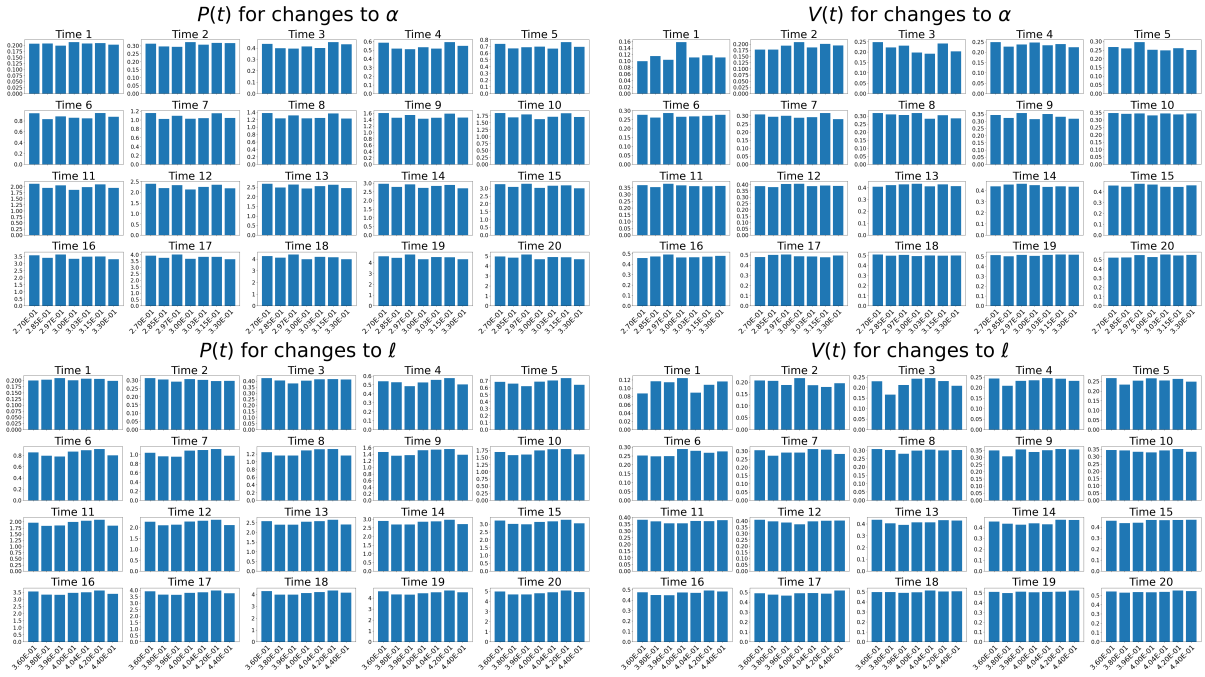

Figure 4.10: Trench plots for α (top) and ℓ (bottom) from data where $\sigma = 10^{-1}$ demonstrating a complete lack of correlation.

expected to move if no other forces act upon it. Specifically, nearly a third of particles move more than half a unit in each direction over a full experiment with a time span of $[0, 100]$, as the standard deviation of the distribution is $\sqrt{0.05^2(100)} = \sqrt{0.25} = 0.5$ and 31.73% of particles will have moved more than one standard distribution.

By choosing $\sigma = 0.05$ for our SDE model simulations we obtain a realistic amount of movement due to random walks without causing the noise in the model output to overcome the observable behaviors caused by the model parameters. In summary, we now have two

particle motion models, one deterministic system of ODEs and one stochastic system of SDEs with random walk noise, which we will simulate using numerical integration over the time interval $[0, 20]$. We justify these dataset simulation choices using statistical measures of correlation between model parameters and metrics that summarize the position and velocity of the population of particles over time.

4.4.3 Summary Functions

In order to estimate the parameters of a mathematical model, we need a method of comparing the ground truth data to the data created by the model using a given set of parameters. Unless we intend to directly compare the outputs of the model to the same quantities in the data, the first step in this process is choosing a function that can be applied to the model output and data which summarizes their important features. For any given problem, there may be hundreds of functions that summarize or otherwise transform raw data. For many-agent models with concrete real-world significance—like our collective particle motion model—it may seem obvious to directly compare the properties of each agent between the true data and model data, an approach we have taken with the above position and velocity metrics. This approach, while straightforward, has its limitations.

The primary issue with these direct-agent comparisons is that they require the same properties for each and every agent in a true dataset. While it is free to obtain this information from simulated ground-truth datasets, if we wanted to use the metrics introduced in the previous sections on real particle motion data we would need the exact positions and velocities of all of the particles in the dataset accurately assigned to the same particle at each time point. Obtaining this kind of labeled real data is often completely infeasible, as discussed in Chapter 2 for the case of tracking cells in microscopy videos. Another relevant downside to these comparison methods is their inability to capture emergent behaviors. Because they only measure local, agent-scale properties, these metrics have no way to directly evaluate large-scale properties of the system, such as the population phenotypes discussed in previous work. Other related limitations of direct-agent comparison include an inherent sensitivity to inaccuracies in the true data, and a high dimensionality or number of values per dataset.

We address these issues by leveraging *summary functions* which reduce the dimensionality of the measures being compared, require less specific data properties, and aim to capture emergent behaviors. While almost any function applied to data can be said to transform that data, we focus on functions that map from a higher-dimensional space to a lower-dimensional space and supply helpful quantitative properties of the underlying

data. For our purposes, we define a *summary function* f mathematically as any function whose range has a dimension less than its domain:

$$f : X \mapsto Y,\ \dim(Y) < \dim(X). \tag{4.14}$$

Though this requirement is rather concrete, even for topological spaces that are more general than the Euclidean spaces we most frequently employ, the stipulation that summary functions yield "helpful quantitative properties" requires subjective interpretation. For example, while the function mapping any n-by-n matrix to 0 certainly results in a reduction of dimension, there is little argument for this providing any useful information about the input matrix. Because we understand the challenges of estimating agent-based model parameters, we are able to find summary functions that do in fact supply relevant information about our point cloud data.

We base our novel summary function on the crocker matrix [97]. By leveraging TDA, we expect to encapsulate multiple scales of behavior and reduce susceptivity to noisy observations while utilizing imprecise properties that can be more reasonably acquired from real data. We also introduce a measure of particle density to be used as a comparison to our crocker-based summary, as it has been successfully implemented in parameter estimation for ABMs by the lab of Dr. Matthew J. Simpson [15].

Crocker Summary Function

We use the vectorized crocker matrix, described in the crocker follow-up article [98], as the basis of our point cloud summary function. To briefly reiterate, the kth crocker matrix is a 2-dimensional array where the elements are kth Betti numbers of a simplicial complex formed from a dynamic point cloud dataset. The rows of the crocker matrix as defined in previous work correspond to different values of the proximity parameter ε, and the columns of the matrix are the time steps of the dynamic dataset. We refer to individual columns as Betti curves, and thus conceptualize the crocker matrix as multiple Betti curves concatenated over time. This particular orientation of rows and columns is not significant outside the convention of visualizing time on the horizontal axis, so to simplify certain steps of our parameter estimation, the orientation is transposed in the output of the operators that follow. There are several predetermined factors needed for the computation of the crocker matrix: the list of proximity parameters $(\varepsilon_1, ..., \varepsilon_n)$, the Betti number k, and the list of time points $(t_1, ..., t_m)$. We refer to these as the *hyperparameters* of the crocker matrix, using this common machine learning term to distinguish these values from the model parameters.

We have already determined the list of time points we will use in earlier sections, but note that different considerations may be needed for real datasets. For example, in an experimental dataset it may be necessary to discard the first few time points due to excessive observation noise or dissimilar dynamics. Therefore we must choose both a Betti number and a list of proximity values that suit our dataset and desired outcome.

Both theoretically and computationally, higher-degree Betti numbers are more difficult to calculate [132]. Furthermore, the Betti number of a given complex is identically zero after a certain degree [133], so while higher Betti numbers may be thought of encoding richer, higher-dimensional information, they also are likely to encode less information, leading to a problem of diminishing returns. Thus we concentrate on lower-degree Betti numbers, in particular choosing the 1st Betti number b_1 which counts the number of 1-dimensional holes, as it strikes a balance between computational tractability and topological significance. While it is of course possible to use more than one Betti number to compute multiple crocker matrices for any given dataset, we utilize only one b_1 crocker matrix so that our observations are one-dimensional and can thus be more fairly compared to the density summary function.

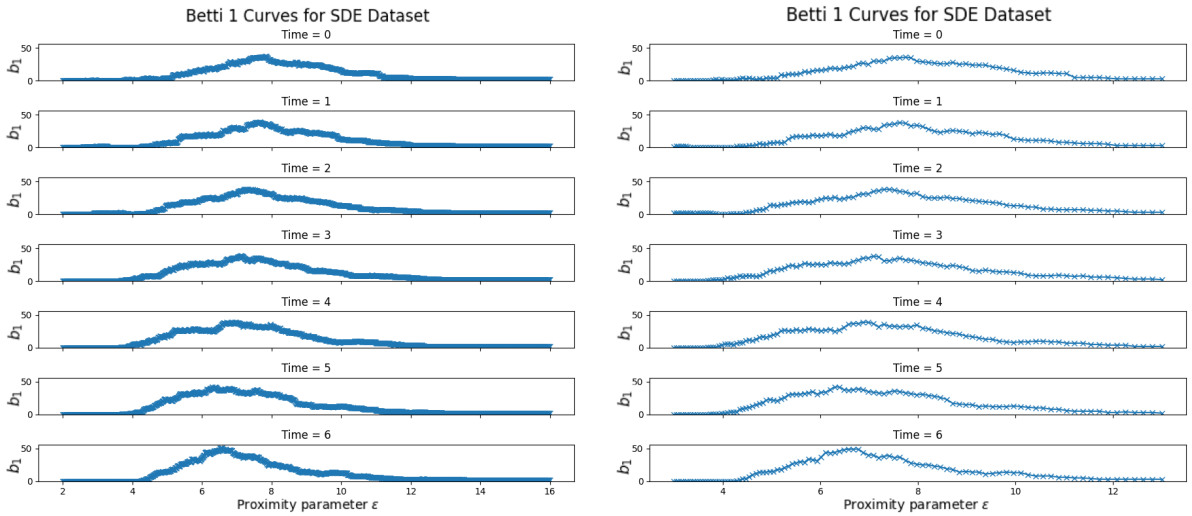

Figure 4.11: Plots of b_1 Betti curves for the first few frames of a simulated stochastic dataset. The curves constructed using only 100 exponentially-spaced proximity values (right) capture most of the changes visible in the plots using 1000 equally-spaced values (left).

In order to choose appropriate proximity parameter values we examine the individual Betti curves of the first frames of a simulated dataset. We first plot the curves using

an abundance of evenly-spaced proximity values, being sure to use a wide enough range so that all non-zero Betti numbers are included (Figure 4.11; left). Rather than using as many values as possible, however, we would like to use fewer, more significant values in order to reduce the size of our summary metric with respect to the original number of data values. Seeing that changes and fluctuations happen more frequently at smaller values of ε, and following the example of the Bhaskar paper [117], we choose to space the proximity values exponentially so that there are more small values than large ones. We also narrow the span in an attempt to eliminate as many identically-zero Betti numbers, which carry no information. Specifically, we take 10 to the power of each number on the linear grid from $\log_{10} 3$ to $\log_{10} 13$, resulting in exponentially-spaced numbers with a base of 10 between 3 and 13 (Figure 4.11; right).

In general, we may rigorously define the crocker function as an operator $\mathbf{C}_k$ on a sequence of point clouds—finite metric spaces—indexed by time $S = (S_t)_{t=t_1}^{t_m},\ S_t \subset \mathbb{M}$ for some metric space $\mathbb{M}$. The operator is parametrized by the sequence of proximity parameters $\mathbf{E} = (\varepsilon_1, ..., \varepsilon_n)$ used in the computation of the Vietoris-Rips complexes.

$$\begin{gathered} \mathbf{C}_k : (\mathbb{M}_i)_{i \in I} \mapsto \mathbb{Z}^{m \times n}, \\ \mathbf{C}_k \left(S; \mathbf{E}\right)_{i,j} = \text{Betti}_{VR}^k \left(S_{t_i}; \varepsilon_j\right), \\ i = 1, ..., m,\ j = 1, ..., n \end{gathered} \tag{4.15}$$

where $\text{Betti}_{VR}^k(X; \varepsilon)$ gives the kth Betti number (b_k) of the Vietoris-Rips complex on the finite metric space X formed with proximity parameter ε. Note that this Betti operator may be further decomposed into expressions involving the metric d on the point clouds, the Vietoris-Rips operator $V\ R$, the number of points N, and the homology functors from algebraic topology. Since these concepts have already been carefully introduced, we utilize this compact notation as it is fully descriptive and unambiguous.

Thus, for our 4-dimensional point cloud datasets and using the aforementioned hyper-parameters, we therefore define our specific summary function $\mathbf{C}_1$:

$$\begin{gathered} \mathbf{C}_1 : (\mathbb{R}_t^{4 \times N}) \mapsto \mathbb{Z}^{20 \times 100}, \\ \mathbf{C}_1 \left(M; (\varepsilon_1, ...\varepsilon_{100})\right)_{i,j} = \text{Betti}_{VR}^1 \left(M_{t_i}; \varepsilon_j\right), \\ t_i = i = 1, ..., 20, \\ \varepsilon_j = 10^{\left[\log_{10} 3 + (j-1)\frac{\log_{10} 13 - \log_{10} 3}{99}\right]},\ j = 1, ..., 100, \end{gathered} \tag{4.16}$$

where $M = (M_{t_i})_{i=1}^{20}$ is the time series of point clouds. Because we do not further investigate the effect of changing the proximity parameters, we assume the use of the

given sequence of ε's and will most often write $\mathbf{C}_1(M)$ to denote the b_1 crocker of M as stated above.

Density Summary Function

In order to compare the accuracy of our TDA-based summary function to an established methodology, we compare it to a summary function used in previous work which is based on the *density profiles* of the point clouds. This method of summarizing particle motion based on a discrete approximation of the population's density is introduced in a 2016 article by Wang Jin and colleagues as a way to study cell migration models [134]. It has since been utilized in other works with the lab of Dr. Matthew J. Simpson [135], including in study that fits agent-based models to experimental scratch assay data [15], a task rather similar to our own. Thus we formalize this cell density profile as a summary function to be used on our simulated data.

To compute the density profile of a single frame of particles, we first divide the domain into a fixed number of evenly-spaced rectangles spanning one of the spatial dimensions. Because the area of each of these rectangles is predetermined, we can estimate the particle density of that rectangle by counting the number of particles contained within it and then dividing that number by the area. By doing this for each rectangle we obtain a profile of density estimates in terms of the spatial dimension that the rectangles span, providing a summary of the density of the entire population.

Thus we can express the density profile p of a population of particles as a function of continuous spatial position x using discrete rectangle locations i: $p(x) = N(i)/(wL_y)$ where $N(i)$ is the number of cells in the rectangle at position i, w is the width of the rectangle, and L_y is the height of the domain (Figure 4.12). Just as the crocker matrix is simply the concatenation of the Betti curves of a sequence of point clouds over time, we base our density summary function on the concatenation of density profiles over time. Thus, to form a matrix that encodes the density of an entire dynamic point cloud dataset, we concatenate the discretized density profiles:

$$\mathbf{P}_{*,j} = p(x_j),\ x_j = x_0 + j\Delta x. \tag{4.17}$$

In order to reformulate this in terms of an operator on a point cloud of particle positions, let us first define an indictor function for membership in a rectangle that is defined by the absolute positions of the rectangles edges. Note that we discretize the

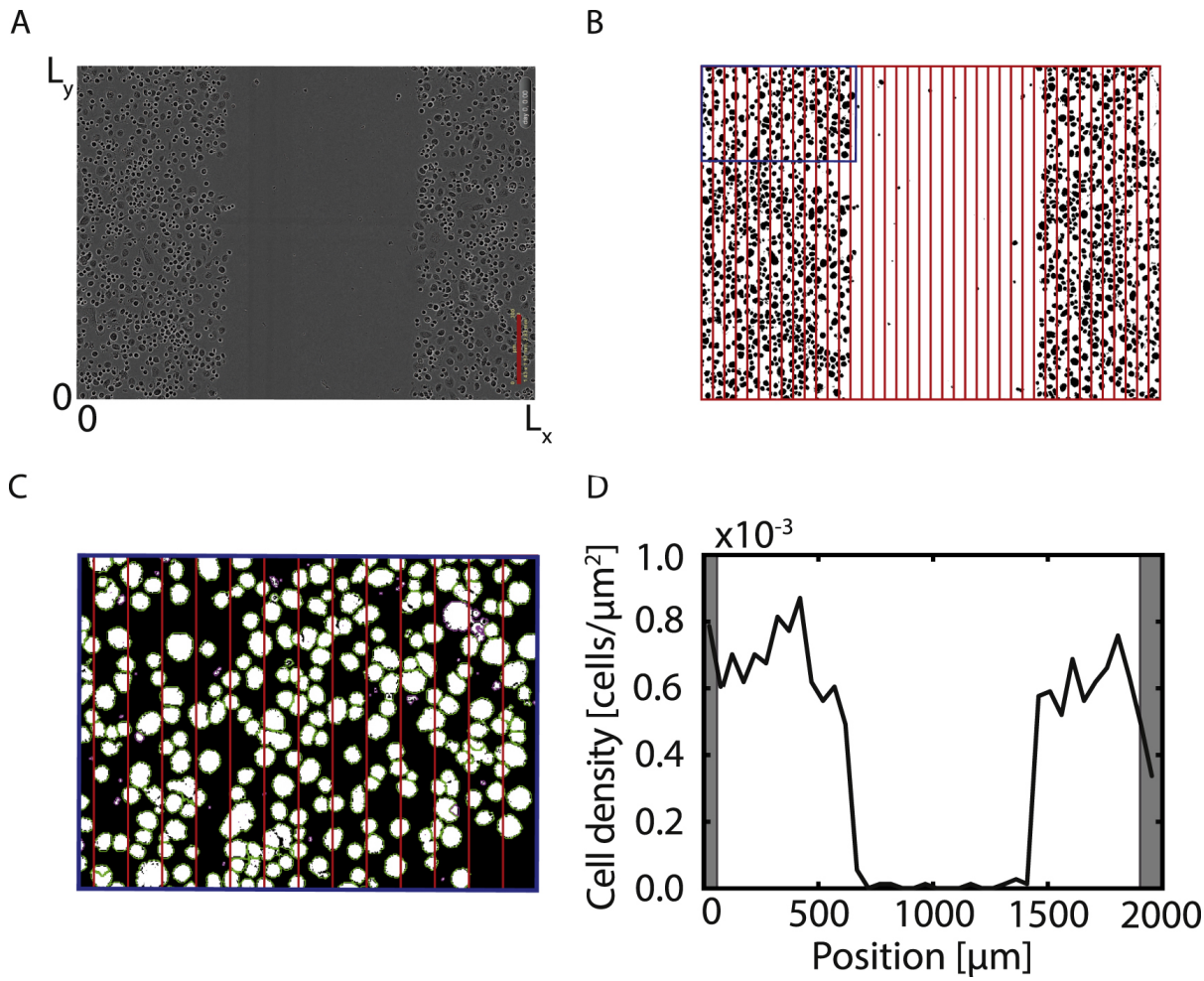

Figure 4.12: A diagram showing the computation of the cell density profile of a single scratch assay image from the work of Matsiaka et al [15].

spatial domain in terms of the vertical position rather than horizontally.

$$\mathbf{1}(\vec{x}; y_{min}, y_{max}) = \begin{cases} 1 & \vec{x}_1 > y_{min} \wedge \vec{x}_1 \leq y_{max} \\ 0 & \text{else} \end{cases}. \tag{4.18}$$

Then we select a sequence of $n+1$ cutoff positions, which serve as the edges of the n rectangles, to be used as the parameters of the above function. For our data we will simply split up the spatial domain into 100 rectangles with width 1 and height 100, so this sequence of cutoffs is $(y_0, ..., y_{100}) = (0, 1, 2, ..., 100)$. Since the area of each of these rectangles is identically 100, we do not divide the sum of the particle indicators by the area. This causes our density function to output integer-valued matrices, and make the elements more directly comparable to those of our crocker function. Note that this makes

it not a measure of density, but of counts of particles. This clearly does not change the use of the function in practice, but makes it easier to discuss the theory of the two operators in subsequent sections. We finally define the density (cell count) summary function as an operator $\mathbf{P}$ on a sequence of position point clouds:

$$\begin{aligned} \mathbf{P} &: (\mathbb{R}_t^{2\times N}) \mapsto \mathbb{Z}^{20\times 100}, \\ \mathbf{P}(M;(y_0,...,y_{100}))_{i,j} &= \sum_{k=1}^{N} \mathbf{1}(M_{t_i}(k); y_{j-1}, y_j), \\ t_i &= i = 1,...,20, \\ y_j &= j = 1,...,100, \end{aligned} \tag{4.19}$$

where $M_t(k)$ is the position vector ($\vec{x} \in \mathbb{R}^2$) of the kth particle at time t.

Robustness to Noise

While many properties of our summary functions are immediately apparent, the fact they are both robust to small changes in the data is less obvious. By this we mean that sufficiently small changes in the raw output of our model—the positions and velocities of the particles—will not change the output of the summary function. This is a result of both the construction of the operators and the fact that they are *integer-valued functions*. Due to the fact that they enumerate a quantity, either by counting the number of k-dimensional holes in a simplicial complex or by counting the number of particles in a subsection of the spatial domain, the summary output will not change continuously with respect to the continuous model output. While this results in some theoretical limitations due to the assumptions of continuity made by the probabilistic treatment of inverse problems, this inherent robustness is a desirable property, particularly for our noisy stochastic model.

There are other properties that cause these summary functions to be robust to noisy input point clouds. The crocker matrix comes with topological guarantees about its stability and ability to remain unchanged under certain types of transformations. Specifically, the homology groups of the filtration, which the crocker measures, are homotopy invariant, meaning they are unaltered by continuous deformations of the underlying point cloud [82]. This fact can be concretely proven for simplicial complexes like the ones used in the computation of the crocker matrix [136]. These types of deformations are not the most relevant to our specific use case, as the particles frequently move independent of one another, but are still useful in considering the impact of particle adhesion and interaction on the summary matrix. Rather the robustness of the summary is mostly achieved by the use of discrete proximity parameter (ε) sampling. A small change

in the position of the point in 4-space corresponds to a small change in a topological birth or death time (see Figure 3.4). Thus, by using a finite sample of proximity values we assure that these disturbances do not propagate to the actual Betti number unless they are meaningful.

The robustness of the density function is built on similar principles; by dividing the spatial domain into discrete compartments, we reduce the chance that a small change in particle position changes the resulting density. This is especially apparent for movement parallel to the y-axis, as this type of movement will never change the cell-count density values. Even for movement in the y direction, the density profile only changes if a particle moves over the rectangle edges which correspond to the discrete cutoff values. Articles utilizing the density profile discuss and more clearly demonstrate this behavior [15], and Figure 4.12 illustrates this in reference to cell migration (see C, D).

While this robustness is a clear advantage in many cases, an inability to report meaningful change would be an obvious drawback. The balance between sensitivity and specificity for these functions is almost entirely dependent on the choice of hyperparameters as described above. Therefore, while we do not explore in-depth the impact of changing the parameterization of the summary functions for our experiments, we recommend careful consideration of the problem at hand while choosing these hyperparameters.

Altogether, we have constructed two operators that take an entire sequence of point clouds as input and produce a 2-dimensional matrix of values that summarize important characteristics of the dynamic point cloud data. These summary functions both provide useful quantitative point cloud information in a way that is demonstrably unperturbed by noise in the raw data, while greatly also reducing the number of values needed to describe the point cloud. Therefore, we use them as the final step in constructing our modeling pipeline and will herein develop the notion of the summary output as an endpoint of our simulation study.

4.4.4 Statistical Model

Now that we have ascertained a complete modeling process beginning with model parameters and initial conditions and ending with a matrix summarizing the important properties of the entire simulation, we need a formal method for recovering the parameters from the final data. We can think of the entire process of simulating and summarizing the data as a single mathematical relation between the input parameters and the final matrix, which is precisely the viewpoint of *inverse problems*. Thus we will herein introduce our pipeline as a forward problem, which will allow us to consider the inverse problem and

produce a concrete method for finding model parameters given data.

The Forward Problem

The ability to study an inverse problem is based entirely on the statement of a suitable forward problem which directly relates the quantity that will be observed to the quantity that will be modeled. Given a manifold of all possible parameter estimators Θ which we call the *model space* and a manifold of all possible observations $\mathcal{D}$ which we call the *data space*, we most generally define the forward problem as the relation that maps a point in the model space $\theta \in \Theta$ to a point in the data space $\mathbf{d} \in \mathcal{D}$ [113]. These spaces are *probability spaces*, such that each of the elements of the σ-algebra on the manifold take values according to the *probability measure*. Because we are not comparing the actual positions and velocities output by the particle motion model, but are instead comparing the values of the matrices output by our summary functions, the data space will consist of the latter. Thus our forward problem will be some mapping $\mathbf{g}$ from random variables that estimate the free parameters $\theta = [\alpha, C, \ell] \in \mathbb{R}^3$ to a random variable whose realizations are summary matrices $\mathbf{G} \in \mathbb{Z}^{20\times 100}$.

Since we have fixed the summary function hyperparameters as previously discussed, and chosen to use pre-determined initial conditions for the particle motion model, we can unambiguously define the summary output matrix as a function of the aforementioned parameter estimator θ:

$$\begin{aligned} \mathbf{C}_1\left(M(\vec{x}_0;\theta);\mathbf{E}\right) &\cong \mathbf{C}(\theta), \\ \mathbf{P}\left(M(\vec{x}_0;\theta);(y_0,...,y_{100})\right) &\cong \mathbf{P}(\theta). \end{aligned} \tag{4.20}$$

Therefore we can capture the entire modeling pipeline of simulating a point cloud and computing its summary matrix with the above notation, and equate our forward problem $\mathbf{g}$ with our redefined summary operators depending on which summary we are using: $\mathbf{g}(\theta) := \mathbf{C}(\theta)$ or $\mathbf{g}(\theta) := \mathbf{P}(\theta)$ for all $\theta \in \Theta$.

Because we are performing a simulation study, we need some way to distinguish the "real" summary matrix that exists in the data space from the output of the forward problem. Using standard parameter estimation notation, we will say that θ_0 represents the *true* parameters used to create the *ground-truth* dataset which exists in the data space. Because this must be defined as a random variable, we say that θ_0 is the estimator that takes on the true values of the parameters in question with probability 1—a *degenerate* random variable. Furthermore, we say that $\hat{\theta}$ represents the *estimated* parameters at any point in the estimation process such that $\hat{\theta} \in \Theta$. Thus, following Tarantola, we state

the forward problem as the "theoretically error-free" relation between the ground-truth summary function and the estimated summary function [137]. Finally, because our two summary functions are identical in dimension, and because we subsequently use the same methods and analyses, we save space by stating each subsequent relation only once (using $\mathbf{C}$), though it applies to both functions. Therefore our specific forward problem is

$$\begin{aligned} \mathbf{C} : \Theta &\mapsto \mathcal{D}, \\ \mathbf{C}(\theta_0) &= \mathbf{C}(\hat{\theta}), \end{aligned} \tag{4.21}$$

which is simply to say that the summary of the true, observable point cloud dataset is predicted by the summary of the modeled point cloud dataset.

It seems unnecessary to define these operators in terms of probability spaces and random variables, as the forward problem as we have stated it will always act on a degenerate random variable that takes on a value with probability 1. Even in the case where our forward operator has a nontrivial probability distribution, like when we use the stochastic particle motion model, the parameter "estimator" that we input always takes on the value of the parameter itself. The utility of defining the forward problem in this manner becomes clear when we define the *inverse problem*, as the stochasticity in our data will, in fact, produce non-degenerate parameter estimators.

The Inverse Problem

The idea behind the study of inverse problems is to examine the true mathematical inverse of the forward problem in order to recover a forward relationship between the output data and the input parameters. This is clearly not a simple task, as we lack an analytical solution of even our systems of differential equations and resort to numerical simulations. A probability-theoretic approach allows us to uncover this relation from data space to model space even when the forward problem is particularly ill-behaved due to nonlinearities, discontinuities, and a lack of differentiability. As with many problems in mathematics, the theory is first developed for well-behaved, linear problems, which we show generalize to our inverse problem.

The first step in this process is defining the least-squares criterion. The *least squares* approach to parameter estimation is centuries old, and can be derived using calculus, linear algebra, or statistics [138]. Least squares fitting is the act of fitting a model to data by minimizing the sum of the squares of the *residuals*, which are the vertical distances from the predicted function to the data. The least squares approach is formalized using the theory of linear regression. While our integer-valued summary functions may best

be considered in reference to discrete Poisson regression [139], we continue to consider the continuous least-squares approach, as the recovery of the continuous values that parametrize discrete distributions has been entirely eschewed in the formal mathematical literature. The inverse problem frame of reference is also lacking in that it is based on continuous probability distributions and manifolds. Because our summaries are integer-valued functions, we must reconcile this by assuming that the data space does in fact lie on a continuous manifold, but that we can only sample integer values from the manifold. While this appears to be a strong assertion, this is exactly the type of assumption made in other studies utilizing integer- or rational-valued functions, including those which leverage the same discrete particle density approximation [135][15]. Satisfied by the results of previous work and the underlying theory connecting discrete and continuous probability distributions, we proceed using the typical continuous approach. Thus we utilize the least-squares minimization problem, which is typically solved iteratively using derivative-based Netwon methods, or all at once for over-determined linear systems using the normal equations. However, by forming the forward problem in terms of probability theory we may develop the least-squares criterion without directly employing derivatives or linear algebra.

We forgo the derivation of the least-squares criterion and instead reference the third chapter of the aforementioned Tarantola text [137], summarizing only the points relevant to our investigation. The least-squares criterion for solving the inverse problem is based on the convenient properties of the normal distribution. Specifically, it requires the assumption that all probability densities are Gaussian. We can trivially require this of our model space via box-shaped search spaces for our parameters, which correspond to uniform distributions. For the deterministic model, the outputs are degenerate random variables, which can be thought of as being normally distributed with variance 0, regardless of the dimensionality.

For the stochastic model, we must investigate samples of our summary functions before assuming that the random variables whose realizations are the integer-valued matrices that they output have Gaussian distributions on the data space. Because we know little about these operators in terms of how they propagate error, we simply investigate the errors between a single ground-truth realization of the stochastic model and many sample realizations to approximate the distribution of each of the elements of the errors of summaries with respect to the data space. In short, we simulate the stochastic model 100 times, plot the histogram of errors between the ground-truth dataset and the simulations, and use these histograms to draw conclusions about the general normality of the individual dimensions of the data space. At 2000 elements, the matrices are small enough to visually

examine each element for normal or uniform behavior, but too large to include in their entirety. Thus we include only a small sample of histogram plots here. Inspection of our plots shows that a plurality of matrix elements appear to be unambiguously normally distributed, with a majority of the remaining elements being either identically zero or too narrowly distributed between two or three integer values to be evaluated. There are some anomalies, such as bimodality and occasional outliers, but these could be due to the relatively small sample size or limitations of the investigation itself, and we consider the evidence sufficiently strong to assume normally-distributed element errors. We are unfortunately incapable of using quantitative tests for normality or density estimation, once again due to our use of integer-valued summary functions. Furthermore, we will attempt our parameter estimation methods on our ODE system of equations, which corresponds to the assumption of a degenerate random variable for the matrix output as mentioned above, due to the fact that there is no stochastic noise to otherwise impact the outcome. This will first and foremost provide an assessment of the validity of our model assumptions before testing it practically with these non-trivial outputs.

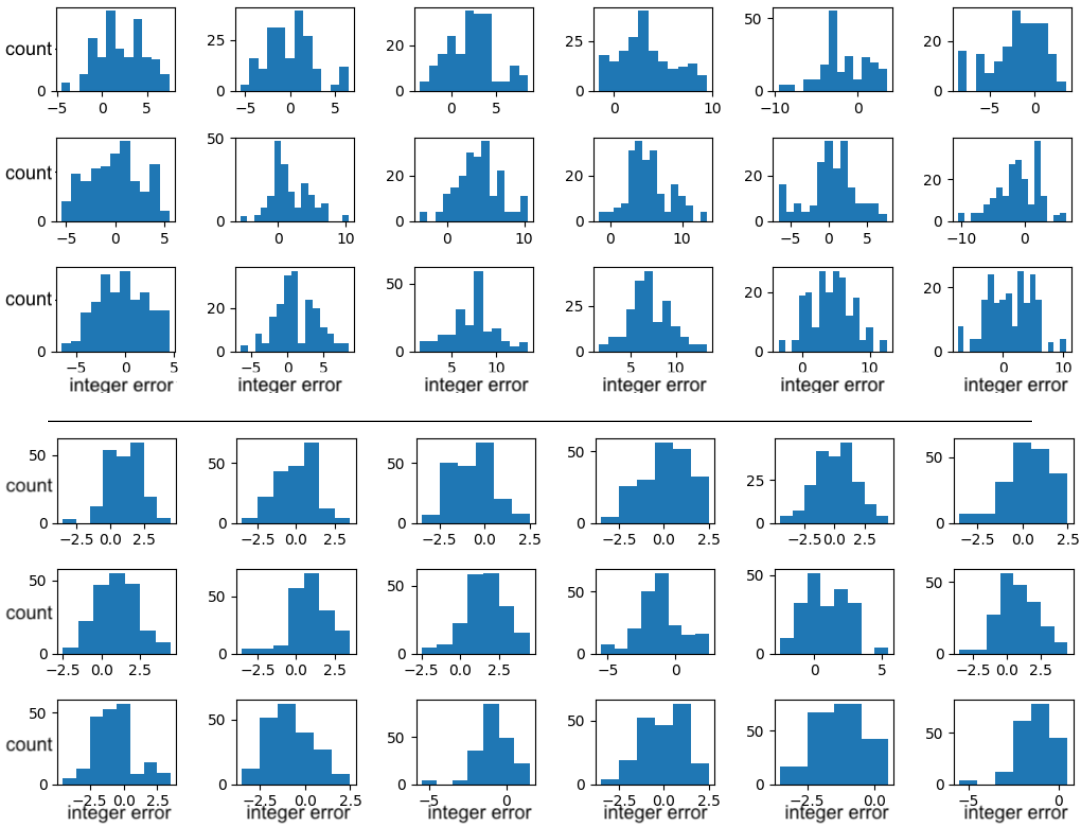

Figure 4.13: A selection of individual element residual histograms from both the crocker summary matrix (top) and density summary matrix (bottom).

The final assumption in the probabilistic derivation of the least-squares criterion is a linear forward problem. While this means that ordinary least squares is in some way best suited for linear problems, the inverse problem derivation is dissimilar from other approaches in that applying the least-squares criterion to nonlinear problems does not completely violate the foundational assumptions. In fact, we may still apply it directly to nonlinear forward problems, we simply have the caveat that these parameter estimators will not be governed by the same guarantees of normality afforded to estimators resulting from linear problems [137]. In practice, the least-squares criterion is almost universally applied to nonlinear problems because most problems are sufficiently well-behaved to yield reasonable results. This approach is referred to as *nonlinear least-squares*, and it is the foundation for the aforementioned derivative-based iterative methods [140]. Yet the probabilistic approach affords us the most general understanding of solving the nonlinear inverse problem, and does not necessitate the use of differentiable functions. Therefore we employ this common ad-hoc approach and use least-squares fitting for our clearly nonlinear and non-differentiable forward problem, electing to examine the normality of the error samples and parameter estimate samples after the fact.

Least Squares Statistical Model

One benefit of the probabilistic framework of inverse problems is that it does not prescribe error to the model prematurely, and does not distinguish between model error and observation error [137]. This is especially useful as we have a nonlinear model whose forward problem we cannot solve analytically. From an inverse problem perspective, this means that we cannot guarantee that the parameter estimates resulting from our least-squares approach will be normally distributed. In other words, we have no way to measure the uncertainty in terms of the error between any given sample from the distribution and the true value without further investigation. While *Bayesian parameter estimation*—the method of obtaining distributions for parameter estimators that are *conditioned* on data via the use of Bayes theorem and likelihood functions—is based on the rigorous treatment of these same error distributions for the sake of quantifying this uncertainty, it is beyond the scope of this work.

Instead we use the standard least squares *statistical model*, which separates the random error from the meaningful model output. We essentially use a linear regression model, treating the data as the sum of a deterministic term and a random error term. While there are a variety of estimators for the solution of this model, we use the ordinary least squares estimator, and subsequently attempt to approximate this distribution of the estimator by collecting a sufficient but tractable number of samples, a simple frequentist

approach. The relationship between inverse problems and statistical models is thus clear, as A. C. Davidson describes statistical models as "a probability distribution constructed to enable inferences to be drawn or decisions made from data" [141].

We consider each element of the crocker matrix to be a separate observation and equally important, leading to the standard, ordinary least squares statistical model

$$\mathbf{C}(\theta_0)_{i,j} = \mathbf{C}(\hat{\theta})_{i,j} + \epsilon_{i,j} \tag{4.22}$$

where $\mathbf{C}(\theta_0)$ is the ground-truth summary matrix resulting from the "true estimator" θ_0, $\mathbf{C}(\hat{\theta})$ is the estimated summary matrix for any $\hat{\theta} \in \Theta$, and ϵ are the unknown error terms for each of the elements of the matrices. We then determine the problem of error minimization by simply solving for the error terms and stating the ordinary least-squares estimator θ^* as the solution of the optimization problem:

$$\begin{aligned} \varepsilon &= \mathbf{C}(\theta_0) - \mathbf{C}(\hat{\theta}) \\ \Rightarrow \theta^* &= \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1,j=1}^{100,20} \left[\mathbf{C}(\theta_0)_{i,j} - \mathbf{C}(\hat{\theta})_{i,j} \right]^2 . \end{aligned} \tag{4.23}$$

This final statement is the minimization of the sum of the squares of the individual errors. Now that we have clearly defined the inverse problem in terms of mean-squared error, what remains is to find a method for solving the resulting optimization problem without the use of derivatives, as we have shown our summary functions to be generally non-differentiable. Once we have done this, the formulation of the inverse problem as a map from random variables corresponding to summary matrices in data space $\mathbf{C}(\theta_0)$ and $\mathbf{C}(\hat{\theta})$ back to the optimal estimator of the parameters θ^* will be readily apparent.

4.4.5 Derivative-free Optimization

Because the standard approach to nonlinear least squares involves taking the derivatives of the objective function with respect to the parameters, we must use a *derivative-free* alternative for minimizing the objective function. Since the slope of the objective surface is not available, most all derivative-free methods rely on *direct search*, which is the methodical examination of many points in the search space with the optimal point based solely on comparisons to other point evaluations [142]. While most direct search methods were developed several decades ago, before the publication of the theoretical results that prove the convergence of quasi-Newton methods, they are still essential for the minimization of non-differentiable objective functions such as ours.

The Nelder-Mead simplex search method was introduced in 1965 as a continuation of previous simplex-based direct search methods [143]. The method works from an initial k-simplex made from $k+1$ points in k-dimensional parameter spaces, conditionally either reflecting, expanding, or contracting each subsequent simplex until an optimum is found. This method is not only more efficient than other direct search methods [142], but has withstood the test of time, being featured as the primary derivative-free unconstrained optimization algorithm for many popular computational packages, such as MATLAB via the "fminsearch" function[144].

Because Nelder-Mead is widely used and performs well on a variety of problems, we use it as the primary method for solving the least-squares minimization problem (4.23). Specifically, we use the implementation available in SciPy's optimize module [145], which uses the same values for the reflection, expansion, and contractions coefficients as the original paper [143]. Our parameter estimation process involves creating a single true dataset using the predetermined model parameters (Table 4.1) and then obtaining a point estimate of the free parameters by using Nelder-Mead to minimize the element-wise sum of squares between the true summary matrix and the summary matrix created using the parameter values obtained at each optimization step. We set the criterion for convergence as the absolute difference between iterations of the parameter values being less than 10^{-3}, and set a maximum of $100k$ iterations, where k is the number of free parameters.

We perform parameter estimation experiments for each parameter individually as well as combinations of multiple free parameters, though the investigation of multi-parameter estimation is limited by our model. In particular, as a consequence of the nondimensionalization process, α and C are coupled in such a way that altering both at the same time results in an *unidentifiable* model, where values of the parameters that differ significantly can provide identical outputs [146]. Thus we only consider the parameter pairs (α, ℓ) and (C, ℓ). In every case we use both summary function models on the same underlying point cloud data with the same experimental setup in order to directly compare their performance. We do this for both the ODE particle motion model and the SDE particle motion model with $\sigma = 0.05$ as detailed above. Altogether, we thoroughly examine the use of these two summaries for parameter estimation of the ABM in question.

The use of a stochastic model with Nelder-Mead optimization does present a bit of a conundrum, as realizations of the stochastic model are random, yet Nelder-Mead is a deterministic method. From the point of view of our inverse problem, parameter estimators exist as random variables from the model space to the real numbers. When using the ODE model with a deterministic dataset, we will always reach the same point

Table 4.2: Search intervals for each of the model parameters.

Par.	Values
α	$[0.01, 0.9]$
C	$[1, 5]$
ℓ	$[0.1, 1]$

at the end of our optimization routine, which represents a point estimate of the free parameter or parameters and a degenerate estimator. When we have a stochastic model, however, where the summary matrix itself is a random variable sampled at each iteration, we are no longer deterministically converging to the mean, but are instead only obtaining a single realization of the parameter estimator θ^*. From an optimization standpoint, we can think of the response surface as constantly and randomly changing; a single "swell" of the response surface can send the initial simplex in a different direction, resulting in convergence to a different local minimum. Since a single sample is never representative of a distribution as a whole, we approximate the distribution of the estimator by repeating the parameter estimation process 100 times per experiment.

Finally, because Nelder-Mead is based on the optimization of simplexes in k-dimensional space, we need a method for assigning the simplex that initializes the optimization process. In practice it is not strictly necessary that the user chooses a simplex, as the original paper suggests a method for initialization using a single point by taking a fixed step from that point in each direction to form a simplex [143]. However, in many cases we are better able to choose a range of values for each parameter than a specific initial guess. In this way we attempt to expand the use of Nelder-Mead to a more comprehensive search by providing initial simplexes based on the maximum and minimum feasible parameter values. This is especially useful for physical and biological models, as given no other information we can still always provide an extreme range of possible values. For example, we can use a range of 1ft to 9ft for the height of an adult. Yet, because the outcome of Nelder-Mead is directly affected by the choice of initial simplex, we need a method for determining the simplex given these parameter bounds. Note that this does not in any way alter the underlying search method, so despite having "bounds" for the parameters the optimization approach is still unconstrained and local.

We thus choose a range of values for each of our dimensionless model parameters (Table 4.2) based on the constraints of "biological realism" outlined in the model paper [116], and introduce a novel method for selecting initial simplexes from the resulting box in k-dimensional space that aims to more exhaustively search the model space. Choosing the intervals for C and ℓ is simple, as we base them on the provided $C > 1$ and $\ell < 1$

constraints and set the other endpoint at an infeasible extreme. Selecting a minimum and maximum for α is more nuanced, so we set these based on reasonable behavior in test simulations. For example, a near-zero value of α results in the undesirable clumping behavior (Figure 4.4) for the previously selected values of C and ℓ, so we choose the smallest value that appears to reduce this behavior at $\alpha = 0.01$. Likewise, at $\alpha = 1$ the interactive effects are completely overshadowed by a particles propensity for self-propulsion, so we choose a value that is near to this.

Choice of Initial Simplex

Now that we have determined bounds on our model parameters, we need a concrete approach to searching the parameter space that results from the taking product of these intervals given any number of free parameters. We thus need a way to search the space by splitting it up into simplexes, as these are the subspaces with which Nelder-Mead works. The only well-cited method for selecting an initial simplex given an interval for each parameter uses pseudo-random placement of each vertex [147]. This does not suit our immediate need, as it is not concrete and relies on performing the optimization with multiple sampled initial simplexes.

Given k free parameters $\theta_i,\ i = 1, ..., k$ and corresponding parameter intervals $T_i,\ i = 1, ..., k$ our parameter domain is a box in k dimensions: $\Theta \supseteq \Omega = T_i \times \cdots \times T_k$. To simplify our investigation, we may map this box to the unit k-cube without loss of generality. In particular we map the minimum parameter values to the origin and the maximum values to the opposing vertex:

$$\begin{gathered}(\min T_1, ..., \min T_k) \mapsto (0, ..., 0),\ (\max T_1, ..., \max T_k) \mapsto (1, ..., 1) \\ \Leftrightarrow \Omega \mapsto I^k = [0, 1] \times \cdots \times [0, 1].\end{gathered} \tag{4.24}$$

All other vertices of the k-cube are similarly the image of the corresponding combination of minimum and maximum parameter values. We need not name this map for use with arbitrary values, as we utilize only the vertices of the hypercube.

From this frame of reference our pursuit of a method for dividing our search space into simplexes becomes a question of *triangulating the hypercube* [148]. This is in fact related to the open question of the "simplexity the hypercube" [149], though we will utilize only the *standard triangulation* of the k-cube [148] rather than searching for a minimal triangulation which is in fact likely to be sub-optimal for Nelder-Mead initialization.

We construct our requirements for the triangulation of the cube by describing the requirements of our initial simplexes in terms of the parameter search. The result is

this standard method of triangulating a k-cube based on combinatorics and simplicial geometry, which can then be mapped back to our parameter intervals. In order to search a satisfactory subspace of the parameter space with each initial simplex, we require that it contains both the minimum of all values and the maximum of all values, which are the origin and the point $(1, ..., 1)$ in our hypercube. We also know that the end result of the optimization is affected by the size of the initial simplex [143], so we also require that the simplexes be of equal size to remove that factor.

Altogether, we require a set of k-simplexes such that each simplex as the same volume, all simplexes contain the vertices $O(0, ..., 0)$ and $F(1, ..., 1)$, and the union of the simplexes forms the k-cube. Fortunately, the standard triangulation of the k-cube meets all of these needs. This standard triangulation A can be constructed in several ways, but we first present the standard construction based on the permutations of the set of k elements [148]. The triangulation A is a set of simplexes indexed by permutation π:

$$A = \left\{ \left[(x_1, ...x_k) \mid x_{\pi(1)} \leq ... \leq x_{\pi(k)} \right] \mid \pi \in S_k \right\} \tag{4.25}$$

where S_k is the symmetric group of degree k. This set of simplexes meets all requirements since the volume of each simplex is identically $1/n!$ [148], and each simplex contains both the origin and its opposite point F, as both $(0, .., 0)$ and $(1, ..., 1)$ satisfy $x_{\pi(1)} = ... = x_{\pi(k)}$ for any permutation π and are thus in every set of vertices forming a simplex.

We may also consider this triangulation as a result of counting the paths between two opposing vertices in the k-hypercube graph [150], but the most natural construction for our application is to think of walking only along the edges of the k-cube, taking one step in each direction. The permutations of the order of these steps gives the collection of simplexes as above. It is in this way that we can be sure we do not lose the equality of volume between simplexes by applying this method to parameters with intervals of unequal length, since each simplex has exactly one edge of length $\max T_i - \min T_i,\ i = 1, ..., k$ with each other edge being the result of Pythagorean triples corresponding to pairs of edges.

For a single parameter we have only a single 1-simplex, corresponding to the interval itself ($[0, 1] \mapsto T_1$), and for 2 parameters we need only the two triangles $[(0, 0), (0, 1), (1, 1)]$ and $[(0, 0), (1, 0), (1, 1)]$ corresponding to the inequalities $x_1 \leq x_2$ and $x_2 \leq x_1$ respectively. The size of the set of simplexes grows factorially, however, as there are $k!$ permutations of k elements, and so the construction of the simplexes becomes more conceptually complicated. The simplex corresponding to the permutation (1,3,2) is shown in Figure 4.14. While the limitations of our particular model disallow the experimental investigation of 3 or

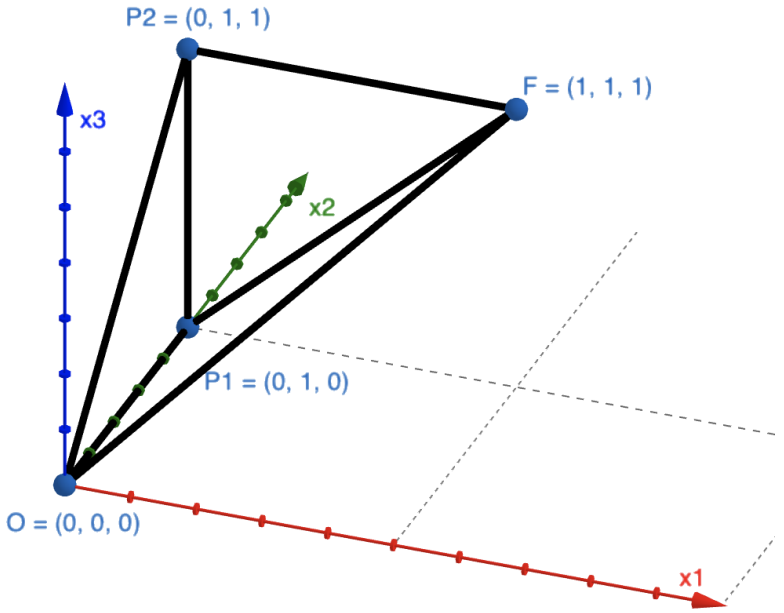

Figure 4.14: One of the simplexes in the triangulation of the 3-cube (I^3).

more free parameters, the underlying theory suggests it is a suitable general approach for exhaustive derivative-free exploration of a sufficiently low-dimensional parameter space, and thus we utilize this simplex selection framework for our 2-parameter experiments and include a general implementation in our software.

4.4.6 Statistical Model Evaluation

Finally, we want to evaluate our choice of statistical model—including our summary function methodology—based on the results of the stochastic experiments. The deterministic model provides little insight into the validity of our statistical model, as we do not recover any measure of uncertainty, but we may use the stochastic model to investigate the propagation of error and the way uncertainty effects the outcome of parameter estimation. In particular, we use the errors between our single ground-truth dataset and realizations of the same random variable to examine the validity of the basic least squares additive error model.

The theory of inverse problems and least squares allows us to make the standard assumption that the error between any element of the summary matrices of any two realizations of our stochastic model takes the form of Gaussian additive noise. By examining the approximate distributions of these errors (see Figure 4.13), we have

confirmed that this was not an unreasonable assumption for our particular investigation. However, we also showed that this assumption was likely violated in certain ways, and that our integer-valued functions limit our examination. The statistician George E.P. Box frequently stated that "all models are wrong, but some are useful" [151], which is exactly the approach we take to evaluating our statistical model. While the subsequent parameter estimation experiment results will show to what end our models are useful, we also want to take reasonable steps into investigating how our models are wrong.

We have a unique ability, due to our use of a simulation study, to examine the actual errors of our model by obtaining many realizations of same true random variable in the form of our summary matrices. Thus we can more directly quantify the limitations of our models, rather than having to rely on the use of residual studies. To this end we will discuss other statistical assumptions for the application of least squares and examine how they are violated, in particular discussing the autocorrelation of our model errors.

4.4.7 Experimental Synopsis

We summarize our entire experimental methodology by saying that we attempt to recover the known model parameters of deterministic and stochastic models with novel topology- and density-based least squares optimization. For each of the Nelder-Mead parameter estimation experiments, we use a single simulated ground-truth dataset to create the "ground-truth" summary matrix for either summary function, and then simulate the model at each optimization iteration to create a single "estimated" summary function matrix. We use 20 time steps and 100 samples per time step in the construction of both types of summary matrices so that they have the same dimensions, and then take the element-wise sum of the squared differences between the data and model matrices as per the aforementioned least-squares criterion and linear regression statistical model. The Nelder-Mead algorithm then attempts to minimize this scalar measure of error with respect to the free parameters, and returns the optimal values of the parameters. For deterministic models, we examine the difference between the parameter point estimates provided by the resultant degenerate estimator θ^* and the true parameter values θ_0. For stochastic models, we must approximate the distribution of the parameter estimator θ^* by collecting multiple estimates. With this sample distribution, we may evaluate the performance of the routine and begin to quantity the uncertainty. Finally, we examine our observable model errors, which are samples from an unknown error distribution, in order to evaluate how well our presumed statistical model corresponds with the true propagation of noise and through the nonlinear operations that we employ.

4.5 Results

4.5.1 Parameter Inference

We report the results of several parameter recovery experiments, including single- and multi-parameter results involving both deterministic and stochastic models. We emphasize the comparison between the two summary models, as our primary goal is the assessment of the topological summary function. Information on the computational cost of these approaches is also included, though we advise that our software implementations are certainly not optimized for speed and thus these costs may absolutely be reduced.

ODE System Experiments

Because the ODE system of equations is fully deterministic, we need only perform the optimization once per experimental condition. There is only one experimental setup for estimating a single parameter as the initial simplex consists of the parameter interval as provided in Table 4.2. For our two-parameter experiments we use 2 separate initial simplexes, selected according to the above hypercube triangulation method. Since there is no noise in this system, our primary aim is to assure that the least squares approach is able to recover the parameters given the complications of nonlinearity and discontinuity. This also helps us to investigate the identifiability and convergence of either summary model with respect to individual parameters.

Table 4.3 shows the results of the single-parameter experiments. The initial 1-simplex and true value are provided along with the point estimate of the parameter. The right-most column shows the signed percent error of the estimate.

Table 4.3: Single-parameter recovery results for the ODE model using density-based loss (left) and crocker-based loss (right).

Par.	**Init.**	**True**	**Result**	**% Err.**
α	[0.01,0.9]	0.3	0.2995868	+0.14%
C	[1,5]	1.8	1.804688	+0.26%
ℓ	[0.1,1]	0.4	0.399707	-0.07%

Par.	**Init.**	**True**	**Result**	**% Err.**
α	[0.01,0.9]	0.3	0.299533	-0.16%
C	[1,5]	1.8	1.8032131	+0.18%
ℓ	[0.1,1]	0.4	0.400480	+0.12%

All of the estimated values are within three-tenths of a percent from the true value, with no apparent dependence on which summary function is used. The two models converge at approximately the same rate given the same convergence criterion, with the crocker model taking an average of 59 iterations to the density model's 58 iterations.

Table 4.4 shows the results of the deterministic two-parameter estimation experiments. The table includes the simplexes used to initialize the optimization, the true parameter value, the estimate, and the signed percent error. We also include the mean of the absolute percent errors, as this is a fair method of summarizing error across multiple parameters. Understanding how the choice of initial simplex affects the end result is the primary goal of this experiment.

Table 4.4: Multi-parameter recovery results for the ODE system using both the density summary model (top) and the crocker summary model (bottom).

Pars.	**Initial Simplex**	**True**	**Result**	**% Err.**	**MAPE**
(α, ℓ)	[(0.01,0.1),(0.01,1),(0.9,1)]	(0.3,0.4)	(0.293,0.144)	(-2.37,-64.1)	33.24
	[(0.01,0.1),(0.9,0.1),(0.9,1)]	(0.3,0.4)	(0.293,0.144)	(-2.37,-64.1)	33.24
(C, ℓ)	[(1,0.1),(1,1),(5,1)]	(1.8,0.4)	(1.799,0.399)	(-0.058,-0.198)	0.128
	[(1,0.1),(5,0.1),(5,1)]	(1.8,0.4)	(1.812,0.399)	(+0.673,-0.357)	0.515

Pars.	**Initial Simplex**	**True**	**Result**	**% Err.**	**MAPE**
(α, ℓ)	[(0.01,0.1),(0.01,1),(0.9,1)]	(0.3,0.4)	(0.2999,0.399)	(-0.020,-0.203)	0.112
	[(0.01,0.1),(0.9,0.1),(0.9,1)]	(0.3,0.4)	(0.2999,0.399)	(-0.020,-0.203)	0.112
(C, ℓ)	[(1,0.1),(1,1),(5,1)]	(1.8,0.4)	(1.742,0.408)	(-3.24,+1.93)	2.585
	[(1,0.1),(5,0.1),(5,1)]	(1.8,0.4)	(2.92,0.313)	(+62.2,-21.7)	41.96

The sensitivity to the initial simplex appears to be entirely dependent on which parameters are free. The performance is significantly different between the two summary function models in both cases, but the (α, ℓ) experiment is not effected by the change in initial simplex in either case while the (C, ℓ) estimations vary significantly.

SDE System Experiments

Our primary result is the estimation of the model parameters using the stochastic model for both the ground-truth dataset and the estimation process. Using a stochastic model for optimization results in a random dataset at each optimization step, resulting in a single realization of the parameter estimator. Figure 4.15 illustrates the distribution of estimates for each individual parameter in the form of histograms, obtained by performing 100 runs of the parameter estimation process for each. This corresponds to the approximate distribution of the least-squares estimator θ^*, so we also report the sample means and standard deviations in Table 4.5. Because we consider the mean of this sample distribution as the point estimate of the parameter, the last column shows the signed percent error of the mean from the true parameter value.

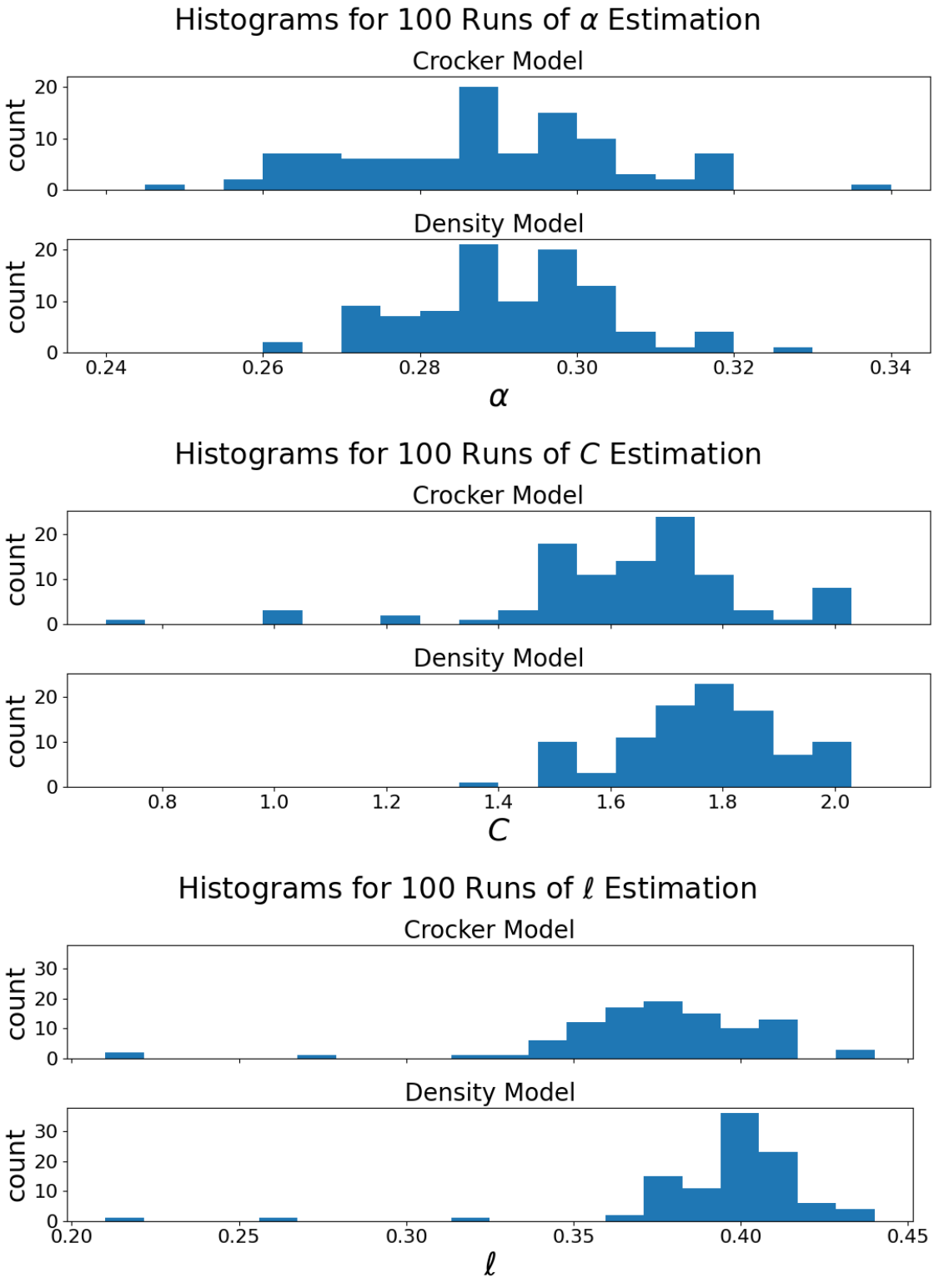

Figure 4.15: Histograms showing the distribution of estimates for each of the three model parameters.

Table 4.5: Parameter recovery results for 100 replicates of the single-parameter estimation process using the SDE model with $\sigma = 0.05$. The left table shows the results for the density summary and the right table shows the results for the crocker summary.

Par.	True	Mean	S.D.	% Err.
α	0.3	0.291565	0.011884	-2.812%
C	1.8	1.761027	0.143425	-2.165%
ℓ	0.4	0.395641	0.027821	-1.090%

Par.	True	Mean	S.D.	% Err.
α	0.3	0.288445	0.016368	-3.852%
C	1.8	1.637004	0.213214	-9.055%
ℓ	0.4	0.409595	0.034441	-6.309%

The error between the mean parameter estimate and true parameter value is never more than 10 percent, yet is always negative, suggesting there is some bias in both models. In fact, the histograms show that the estimate rarely overshoots the true value, and that outliers almost exclusively lie far to the left of the mean.

We also attempt the 2-parameter estimation with 100 replicates of the SDE model. Table 4.6 shows the means, standard deviations, and errors for both initial simplexes for each of the experiments. Figures 4.16 and 4.17 show scatter plots of the estimates along with kernel density estimates of the corresponding marginal single-parameter estimator distributions. These figures demonstrate in greater detail the specifics of the distribution and highlight that, despite differing quantitative results, the two summaries are consistent in that they recover parts of the same distribution.

Table 4.6: Multi-parameter recovery results for 100 runs using the $\sigma = 0.05$ SDE model with both the density summary (top) and the crocker summary (bottom).

Pars.	Initial Simplex	True	Mean	Std. Dev.	% Err.
(α, ℓ)	[(0.01,0.1),(0.01,1),(0.9,1)]	(0.3,0.4)	(0.294,0.331)	(0.013,0.089)	(-2.161,-17.23)
	[(0.01,0.1),(0.9,0.1),(0.9,1)]	(0.3,0.4)	(0.297,0.345)	(0.015,0.085)	(-1.005,-13.87)
(C, ℓ)	[(1,0.1),(1,1),(5,1)]	(1.8,0.4)	(2.576,0.326)	(0.874,0.074)	(+43.13,-18.54)
	[(1,0.1),(5,0.1),(5,1)]	(1.8,0.4)	(1.349,0.462)	(0.330,0.061)	(-25.04,+15.59)

Pars.	Initial Simplex	True	Mean	Std. Dev.	% Err.
(α, ℓ)	[(0.01,0.1),(0.01,1),(0.9,1)]	(0.3,0.4)	(0.301,0.333)	(0.018,0.079)	(+0.487,-16.81)
	[(0.01,0.1),(0.9,0.1),(0.9,1)]	(0.3,0.4)	(0.299,0.342)	(0.022,0.076)	(-0.286,-14.50)
(C, ℓ)	[(1,0.1),(1,1),(5,1)]	(1.8,0.4)	(3.800,0.236)	(1.513,0.080)	(+111.1,-41.10)
	[(1,0.1),(5,0.1),(5,1)]	(1.8,0.4)	(1.084,0.473)	(0.333,0.067)	(-39.77,+18.35)

Table 4.6 illustrates comparisons to the ODE system experiments as shown in Table 4.4. While the (α, ℓ) experiments do appear to be less sensitive to the initial simplex than the (C, ℓ) experiments as before, the differences between models are less pronounced.

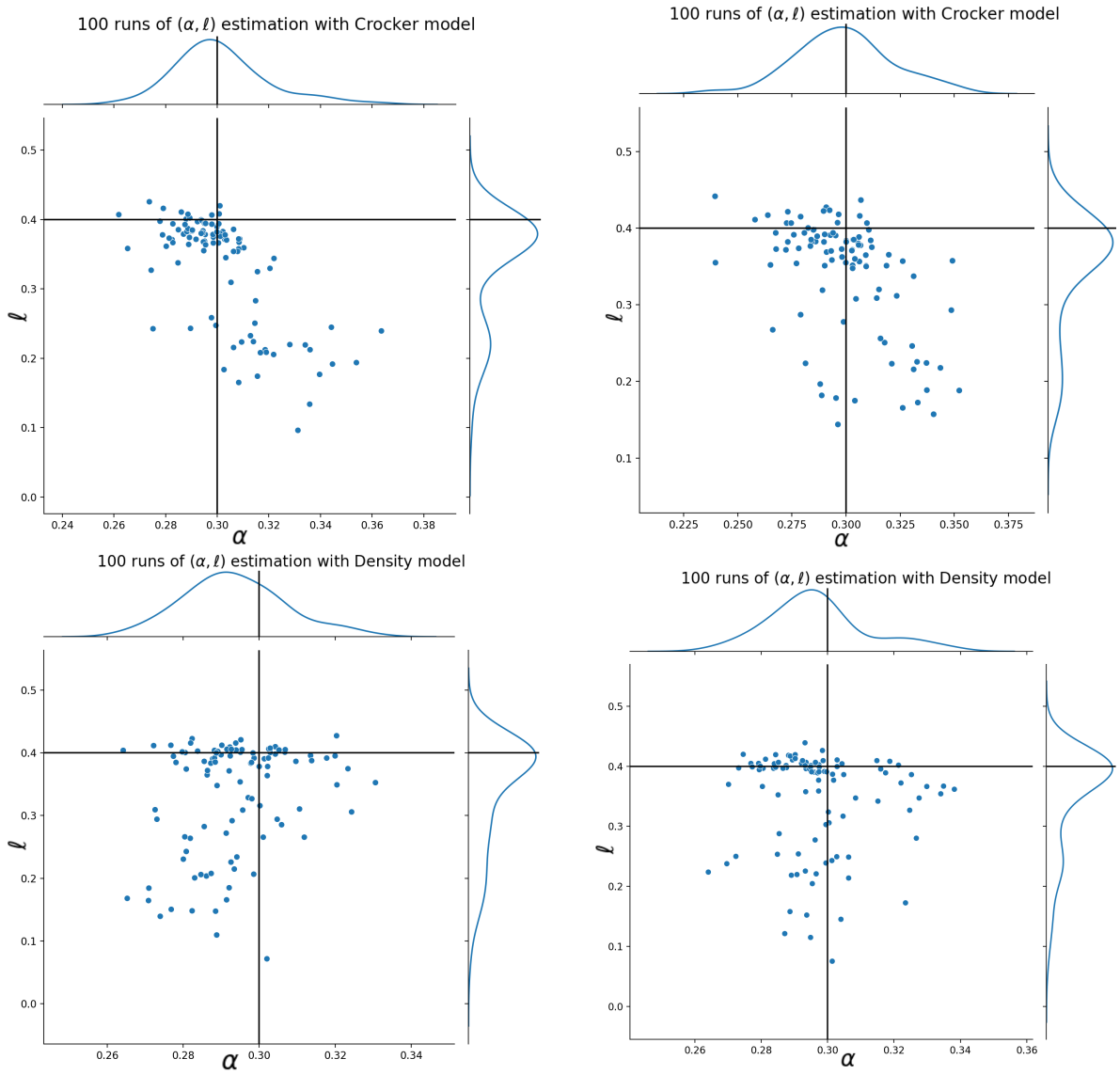

Figure 4.16: Plots of the parameter estimates for 100 runs of the stochastic (α, ℓ) recovery experiment showing the effect of the initial simplex on the distribution of parameter estimates. Black lines mark the true parameter value $(0.3, 0.4)$.

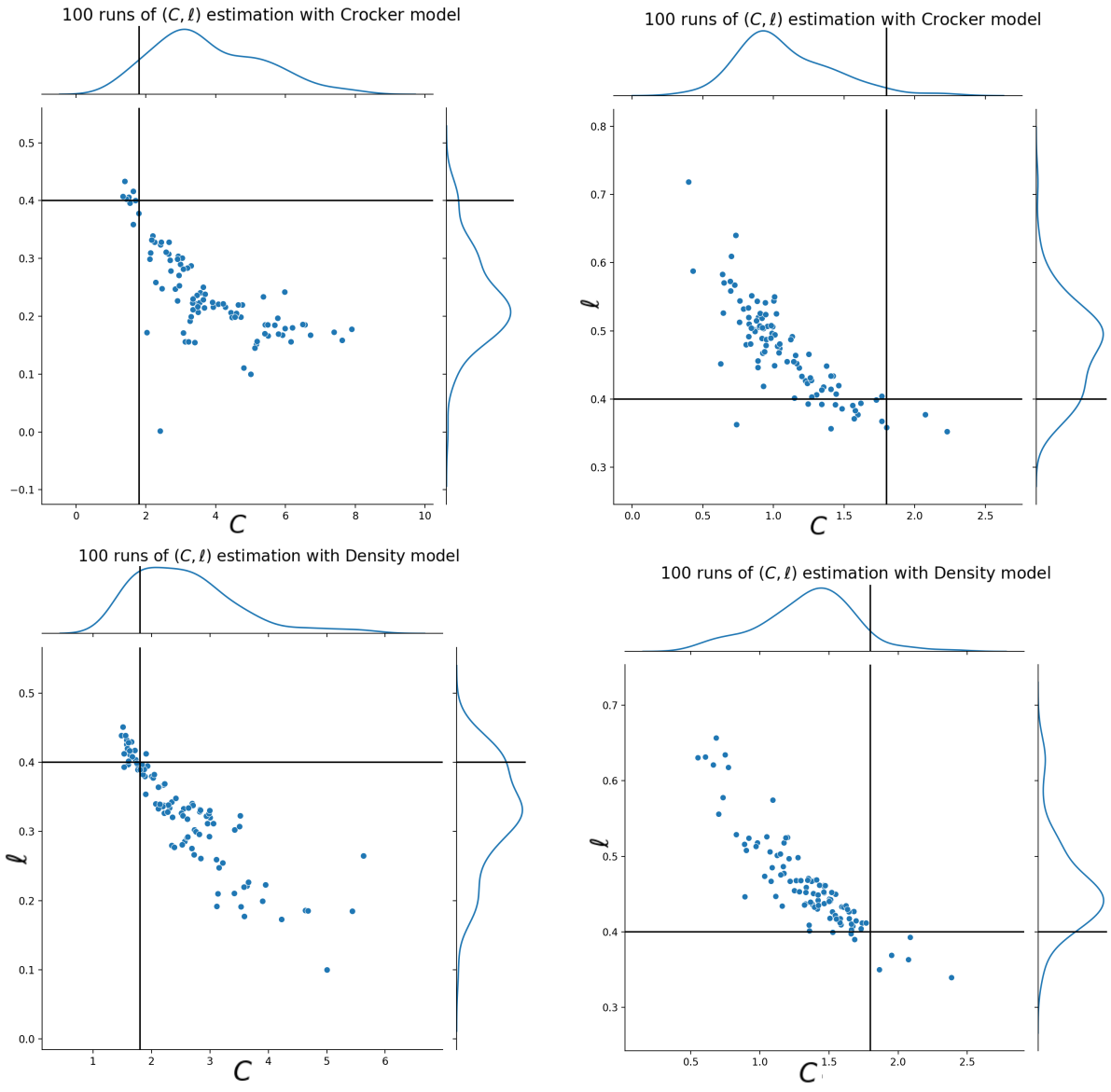

Figure 4.17: Plots of the parameter estimates for 100 runs of the stochastic (C, ℓ) recovery experiment showing the effect of the initial simplex on the distribution of parameter estimates. Black lines mark the true parameter value $(1.8, 0.4)$.

Computational Cost

While the code we use for these investigations is by no means designed to minimize runtime, we may still discuss in general terms the complexity of the algorithms behind these procedures and provide relevant practical requirements. The crocker summary function is clearly more conceptually complicated than the computation of density profiles, yet, due largely to the topological optimizations implemented in Ripser [99], it is only trivially more complex than a naive density calculation for a multithreaded processor. The reasons for this are largely based on the memory demands of persistent homology computations, which are addressed in the literature. However, both of these tasks are *embarrassingly parallel*, meaning they can be trivially divided into many separate tasks. The density computation can be performed for each rectangle in parallel, and there have already been parallelized implementations of the Ripser algorithm [152].

In practice, a full optimization iteration, consisting of either 2 or 3 particle motion model simulation and summary function calculation combinations, takes between 1 1/2 and 2 minutes on a high-powered computing cluster with 2 cores per process. There is little variance in this timing based on model parameter, but for multi-parameter experiments the crocker function timing is significantly quicker, taking an average of 1.62 minutes compared to the density function's 1.72 minutes. This timing is significantly improved on a local machine where access to resources is less limited. On a 2017 MacBook Pro, we measure an average of 3/4 of a minute per iteration. While this is reasonably tractable for single estimation experiments, these computational demands quickly grow beyond the scale of local resources. For example, our 100-run SDE experiments utilize 200 separate cluster cores for the sake of performing all runs in parallel.

4.5.2 Model Evaluation

The plots in Figure 4.18 show the bias in errors by plotting the sample means of the errors for each element. Not only are there many elements with strongly non-zero biases, but there is clear structure to the matrices, showing that there is correlation between errors across elements of the matrices.

Figure 4.19 shows a plot of the mean errors for the crocker summaries. These mean errors demonstrate clear patterns of *autocorrelation*, where errors at future times depend on past time points, and *heteroscedasticity*, a general increase in variance over time. These patterns are consistent with the presumed correlation between errors shown in Figure 4.18.

While the corresponding mean error plot is not possible for the density summaries

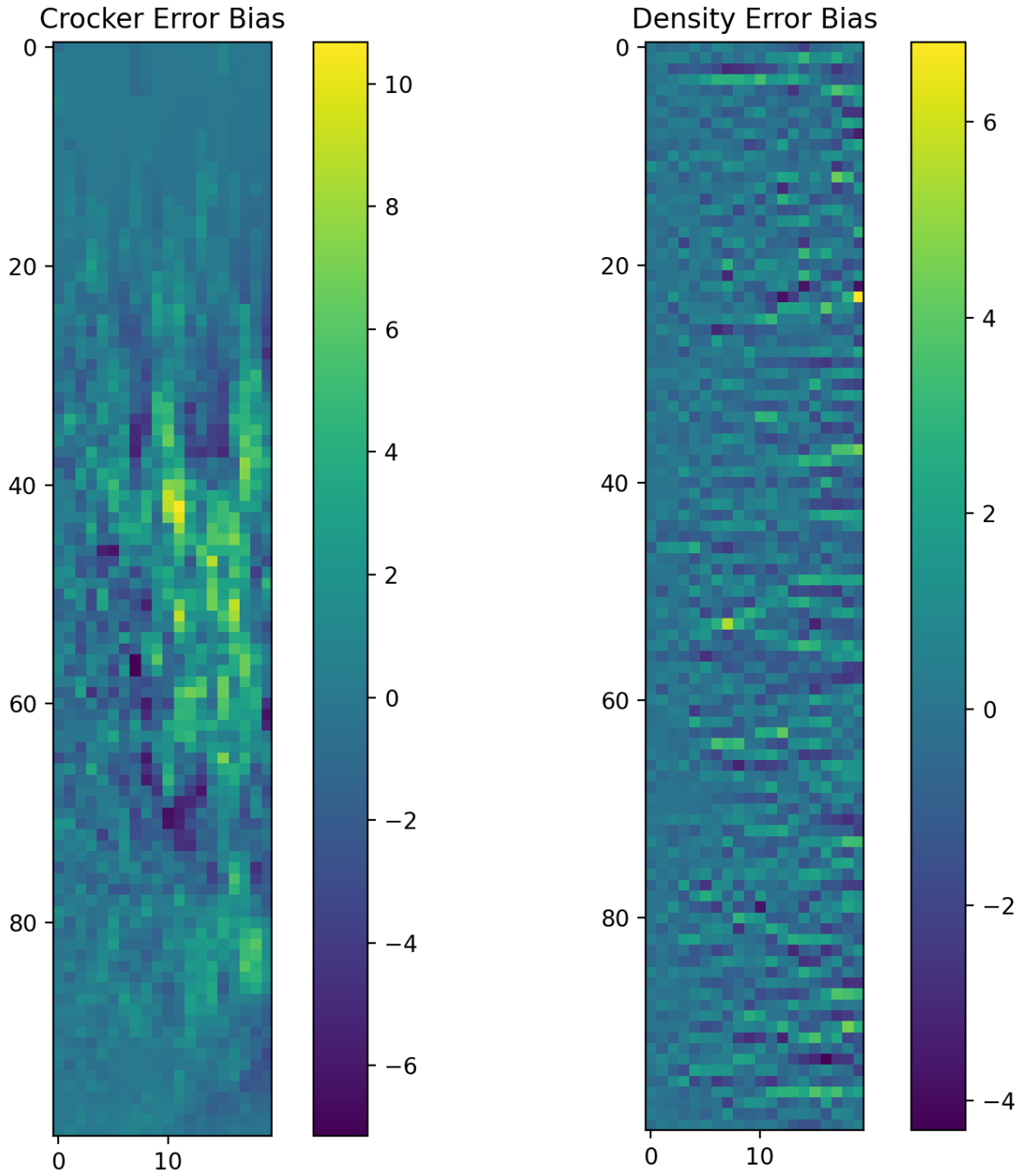

Figure 4.18: Matrix plots showing the sample bias of each of the error terms for both summary functions.

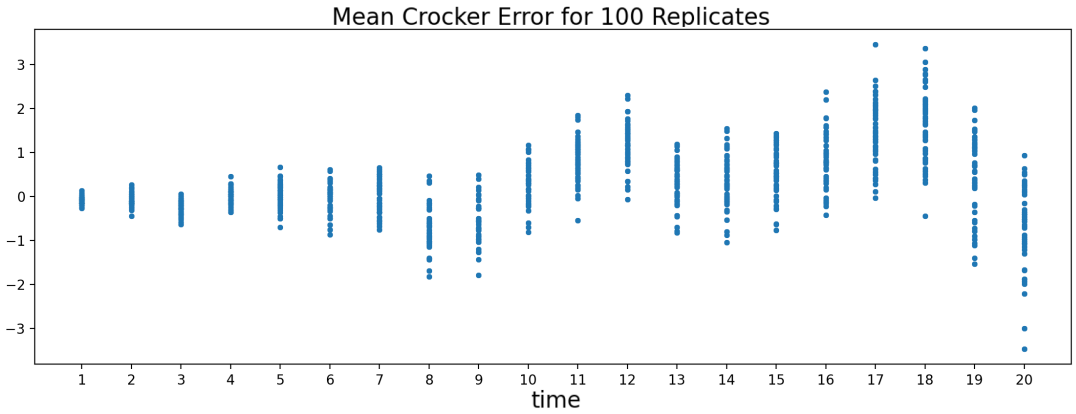

Figure 4.19: Plot showing the mean error over the columns of the crocker summary for 100 replicates plotted over time.

since the mean particle count for an entire image is always exactly the number of particles over the number of rectangles ($400/100 = 4$), we can examine the same time-dependent patterns by plotting the integer error for each rectangle in each frame on the same figure. This plot—shown in Figure 4.20—is limited due to the integer error values, but still demonstrates the same heteroscedasticity and autocorrelation.

These plots demonstrate precisely the types of errors that lead to generalized forms of the least-squares criterion. Errors with certain structure also suggest the use of different statistical models, which we will discuss herein.

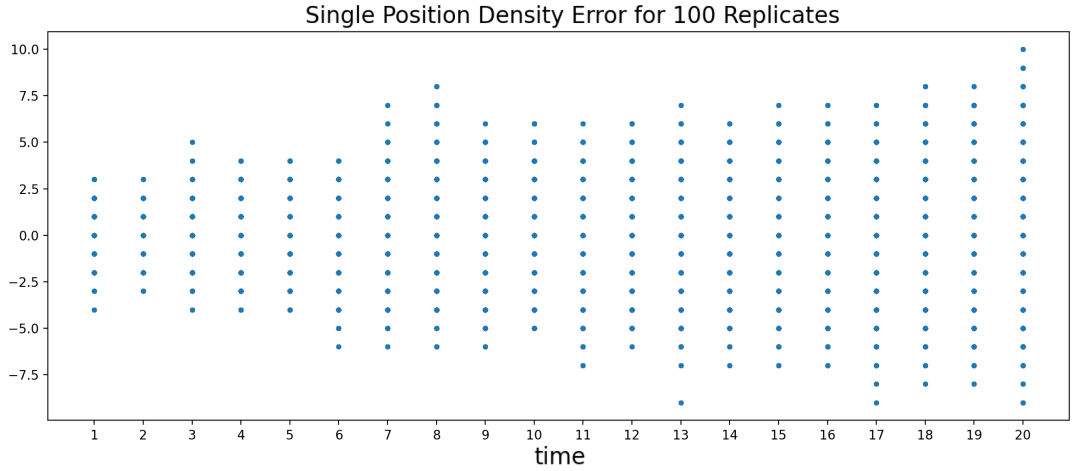

Figure 4.20: Plot showing the errors for each element of each of the 100 replicates of the density summary, plotted over time.

4.6 Conclusions

4.6.1 Discussion

Using a relatively simple least-squares statistical model, we are able to recover the parameters of our stochastic agent-based particle motion model with reasonable accuracy using either a summary of the population density or a summary of the population topology as the forward problem output. While the results do report a notable amount of variance and uncertainty, the point estimates are often within 10% of the true value, and the distribution of estimates obtained from replicate data always suggests that the true value is acceptably contained within the distribution. Specifically, the true value always lies in the *support* of this sample distribution, meaning—in this context—that the convex hull of the estimates always contains the true value, even for the least accurate estimators (see Figure 4.17). The decidedly non-normal distribution of parameter estimates, especially for the multi-parameter problems, is likely caused by the distinct nonlinearity of the forward problem. The ordinary least-squares criterion, when used in conjunction with nonlinear models, guarantees this behavior, and we can likely account for this error with improved statistical model assumptions. We therefore conclude that the true parameter value can be recovered in probability with our parameter estimation method.

The results for recovering deterministic model parameters suggest that our two approaches are equal in terms of single-parameter estimation, but that there are differences in efficacy between the methods for the more-complex multi-parameter estimation problem. Based solely on these deterministic results, we might assume the crocker method is superior based on its ability to reasonably recover any combination of parameters given a good initial simplex. Our stochastic model results, however, cast some aspersions on our crocker model, as the variance of the collection of estimates is generally higher, and the point estimates are almost universally less accurate. The one exception is for the joint estimation of α and ℓ, which—consistent with the ODE system results—obtains better performance with the crocker method than with the density method. This relative reduction in performance is especially concerning given the fact that the crocker model, as we have designed it, requires both position and velocity information, whereas the density model requires only particle positions. Furthermore, the statistical model analysis shows that some of the Betti numbers are identically zero for all sample replicates, suggesting some of our output dimensions carry no information. This can likely be explained by an inaccurate statistical error model, as the crocker summary errors appear less consistently normally distributed than the density summary errors.

This is not, however, a censure of topological approaches to parameter estimation

as a whole. To the contrary, related research has shown the applicability of TDA to inverse problems [121]. Instead, these results illustrate the limitations of our particular preliminary design. In particular, Betti numbers do not provide any information about specific topological features as they only count the number of k-dimensional holes. Other summaries of topology like the *vineyard* [153] and the *formigram* [120] do keep track of individual features, and also exist in metric spaces that make quantitative comparison possible. More meaningful summary functions could be formulated by keeping track of individual topological features of our point cloud, which could in turn result in fewer dimensions with greater interpretable biological significance.

4.6.2 Future Work

The probabilistic treatment of parameter estimation provided by this inverse problem framework does, of course, suggest the use of *Bayesian parameter estimation*. Bayesian parameter estimation is based on applying Bayes' rule to accurately condition a *prior distribution* on the data in order to recover a *posterior distribution* [154]. The end goal of this methodology—to obtain a distribution of parameter values that reflects the observed data—is related to ours, but the means of acquiring a distribution is based on the Bayesian frame of reference that treats model parameters as random variables. Our frequentist approach, in contrast, treats a parameter θ as a fixed, unknown quantity, and attempts to recover a distribution for the parameter estimator θ^* using realizations of the random variable. The primary practical distinction is that Bayesian parameter estimation requires the *likelihood function* of the statistical model. The likelihood function can be viewed a function of the parameters which describes the probability of obtaining the observed data given those parameters [154]. This function definition is what enables the computation of the Bayes factor. As we have discussed above, we do not have a suitably accurate error model, and thus are unable to obtain useful likelihoods.

This leads to the proposition that a better error model could not only lead to better results for our frequentist analysis, but could also permit the use of Bayesian parameter estimation. The most obvious extension of our current crocker summary formulation would be to take the mean of each Betti curve, a quantity we call the *mean homology* of a stationary point cloud. Concatenating these mean homology values over frames of a dynamic point cloud dataset not only gives a single-valued time-series summary of the topology of a dataset, but also brings the crocker summary into the realm of real-valued functions, which is much more appropriate for continuous parameter estimation. Furthermore, we have already shown that the errors of a mean homology model are more

normally distributed, as the plot of mean Betti number errors shown in Figure 4.19 is equivalent to the plot of the errors of the mean homology. Thus one could employ the statistical model

$$\begin{aligned} \overline{\mathbf{C}}(\theta)_j &= \frac{1}{|\mathbf{C}(\theta)_{*,j}|} \sum_{i=1}^{n} \mathbf{C}(\theta)_{i,j}, \\ \overline{\mathbf{C}}(\theta_0)_j &= \overline{\mathbf{C}}(\hat{\theta})_j + \epsilon_j, \end{aligned} \tag{4.26}$$

where $\epsilon_j \sim \mathcal{N}(\mu, \sigma)$ and j indexes the time steps. While the mean homology error plot makes it clear that the errors are not identically distributed, this can be remedied by considering the time steps as T separate dimensions, such that we have a single error term $\epsilon \sim \mathcal{N}(\vec{\mu}, \Sigma),\ \vec{\mu} \in \mathbb{R}^T,\ \Sigma \in \mathbb{R}^{T \times T}$, making this model suitable for Bayesian parameter estimation. Finally, this demonstrates an advantage of the crocker approach, as there is no way to obtain a real, scalar value of a point cloud's density that varies over time.

4.6.3 Contributions

In light of the results and discussion, our primary contribution is the successful application of TDA to the least-squares inverse problem of parameter estimation. While previous publications have used crockers for model selection using experimental data [98] and for clustering deterministic D'Orsogna model data into phenotypes and parameter bins [117], our investigation extends this TDA framework into the recovery of continuous distributions of parameter estimates from stochastic data. This prototypical topological parameter estimation framework shows performance that is relatively similar to that of the equivalent discrete density profile framework, which succeeds at recovering parameters for similar agent-based models [15].

Other developments are made in the course of this parameter estimation investigation. Our formulation of the D'Orsogna particle motion model as a system of stochastic differential equations, while an elementary implementation of Itô calculus stochasticity, is a novel combination of random walk dynamics and the deterministic self-propulsive and interactive forces in the original model. Furthermore, by investigating various measures of error under realizations of this model, we are able to show how this normally-distributed positional randomness propagates as nonlinear noise that is heteroscedastic in time. Also, the method for selecting initial simplexes for Nelder-Mead derivative-free optimization when the parameter search space is defined by a box rather than a point guess by leveraging the standard triangulation of the cube is heretofore unpublished. While this approach does scale factorially, we have shown that these disjoint simplex choices divide the k-dimensional box into a partition. The end result is an optimization method that is,

in fact, sensitive to different simplexes as shown in Figure 4.17 and, to a lesser extent, Figure 4.16.

Chapter 5

TDA as a Measure of Distinct Biological Behaviors

We now bridge the gap between the analysis of noisy simulated data and the investigation of quantified biological data by applying topological data analysis techniques. By examining the topology of both real and synthetic particle motion datasets, we aim to uncover distinct features of otherwise similar populations of interactive particles. We are particularly interested in biological properties that are not explicitly included in the model such as cell morphology, contact inhibition of locomotion, and fluidization.

5.1 Related Work

In order to understand the hypothesized differences between the topology of low-density and high-density cell microscopy data and the differences between these real data and data simulated using a modified D'Orsogna model, we need to better understand some of the complex biological factors that affect the dynamics that are visible in this fibroblast dataset. Additionally, we review previous research that leverages TDA to investigate these biological realities, and see how we may apply this same approach to our novel data.

5.1.1 Cell Migration Biology

Morphology is the biological study of shape and structure, and in cell biology it generally refers to the investigation of the shape of the cells in question and the how changes in cell shape correspond to changes in behavior [155]. Cell morphology is particularly relevant to the study of fibroblasts, as they have unique tendency to be elongated and multi-polar, leading to widely heterogeneous shapes [156]. This morphological behavior is hypothesized

to be a primary factor in the distinctly slow and poorly-understood migration patterns of fibroblast populations [35]. The exact mechanisms of fibroblast migration are complicated enough that they remain an open question in biological research.

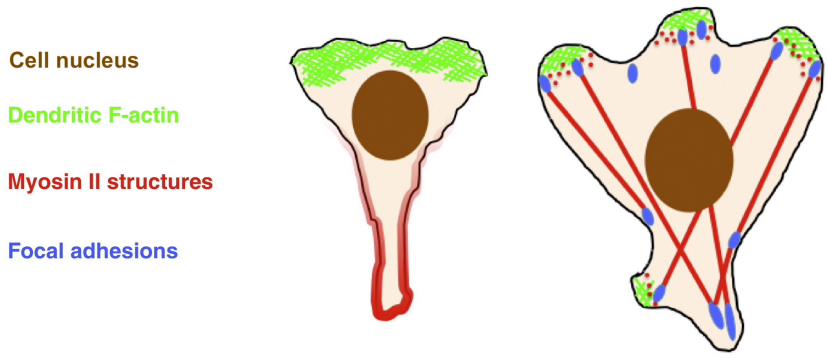

Figure 5.1: A diagram illustrating the complex morphological and adhesive structure of fibroblasts (right) as compared to amoeboid cells (left). Adapted from [35].

Of particular interest is the effect of *contact inhibition of locomotion*, the propensity of cells to change their direction of motion upon contact with another cell, first observed in fibroblast populations nearly 70 years ago [157]. Despite a long-standing knowledge of the existence of this phenomenon, little is understood about how exactly it affects migration patterns. We hypothesize that this effect is more pronounced in the more biologically-realistic confluent or over-confluent fibroblast populations, as there is necessarily more contact between cells.

Even more confounding in the context of this knowledge of contact inhibition is the appearance of cell *fluidization* in densely-packed, over-confluent fibroblast populations. This behavior consists of cells at a sufficiently high density behaving more like a fluid than a population of discrete units, with discernible flow dynamics. While there is research into the fluidization of certain types of biological tissue [158], few publications address this behavior in cell monolayer experiments. Despite being presumably more effected by contact inhibition of locomotion, we often see cells moving through discernible channels, creating shear flows while moving with persistent locomotion. There have been studies into the effect of morphology on persistent migration of individual fibroblasts [159], but this behavior in over-confluent populations remains an apparent contradiction.

While a thorough treatment of fibroblast morphology is beyond the scope of this work, recent research has made clear the myriad factors at work in fibroblast migration [35] [160]. These elements include various types of chemotaxis and interactions with the extracellular

matrix (ECM) in addition to the the aforementioned components of morphological effects. In any case, it is clear that these factors are not all accounted for in most mathematical models of cell migration. While we are able to capture a certain proportion of these effects with more general model components such as interactive forces and friction, we are unable to capture this wide variety of behaviors with such simple modeling approaches.

5.1.2 TDA and Collective Motion

We have already seen how TDA has been applied to particle motion systems, particularly with the crocker plots applied to both the Vicsek and D'Orsogna models [97]. The paper by Bhaskar and colleagues furthered this research by applying crockers to several distinct versions of the D'Orsogna model, and uncovered the topology of the different parameter sets and phenotypes of the model [117]. While these works are of great importance to the application of TDA to collective motion models, these are still limited in their analysis of multi-scale behaviors and their ability to generalize to biological data.

An article by the authors of the original crocker plot work was the first to apply crockers to real biological data [98]. By directly comparing the Betti numbers in the crocker matrices of simulated data to experimental aphid walk data (see Figure 5.2), they are able to confirm the selection of an interactive particle motion model over a standard random walk model with as much statistical significance as the state-of-the-art order parameter approach. This work is fundamental in proving the ability of crockers to evaluate the fit of a stochastic model on genuine experimental data. Importantly, this work leverages a relatively large-scale dataset of hundreds of simulations and thousands of time steps, and involves direct comparison with a ground-truth dataset. While they do discuss important interpretability characteristics, further results on the qualitative evaluation of crockers could provide more information on their sensitivity to finer details and smaller-scale behaviors.

More recent research has further investigated the interpretability of TDA as applied to cell migration assay data [118]. Bonilla and colleagues explore and decipher the topological significance of cell spreading using active vertex model (AVM) migration data. In particular, they illustrate the effect that "islands" of cells have on the persistence diagrams and Betti curves of the VR filtration of the resulting point cloud. This study advances the understanding of qualitative evaluation of cell migration using Betti numbers over separate time steps.

In general, we want to be able to "read" the crocker plots of cell migration datasets and understand the significance of the visible patterns. This body of work provides insight

into how that is possible, particularly for biological collective motion models and datasets whose values lie in some interpretable, physical space. We leverage these ideas and methods in the investigation of our scratch assay data, and consequently summarize how datasets of different densities vary from each other intrinsically and how our candidate D'Orsogna model compares to real cell migration data.

5.2 Methods

The crux of this investigation is the adaptation of varying datasets for the sake of direct topological comparison, and the use of our relatively simple particle motion model to approximate scratch assay cell migration dynamics. We employ a trajectory sampling methodology to manage the confounding effect of varying cell densities, and aim to uncover inherent topological differences between low-density data and high-density data. We also use the same D'Orsogna particle motion models with dimensional parameters to simulate realistic cell trajectory datasets, and compare the topology of these to that of the true scratch assay data.

5.2.1 Particle Trajectory Subsampling

Directly comparing the topology of point cloud datasets of different densities is immediately unreasonable, as a higher-density dataset forms topological features at an entirely different spatial resolution. In other words, a dataset with more points in the same fixed spatial or hyperspatial domain will necessarily have more connected components at smaller proximity parameter (ε) values, more 2-dimensional cycles, and more topological features in general. There are surprisingly few methods for comparing point clouds with different cardinalities—or numbers of points—using topology. The primary method for comparison

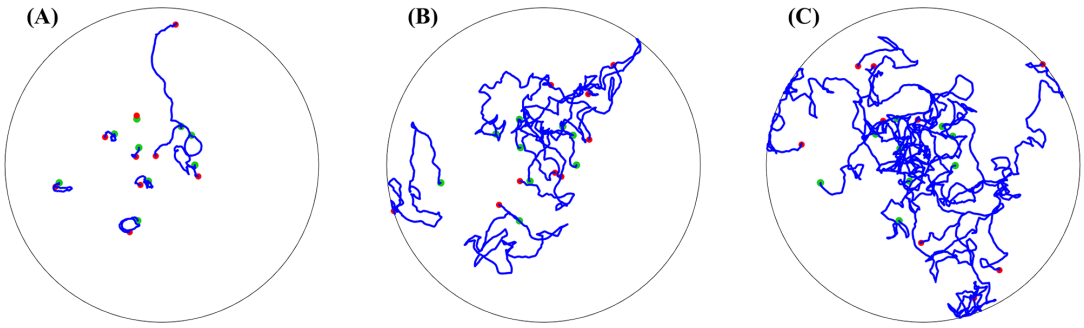

Figure 5.2: Plots of trajectories from the experimental aphid data (A) and from two stochastic walk models (B) and (C) [161].

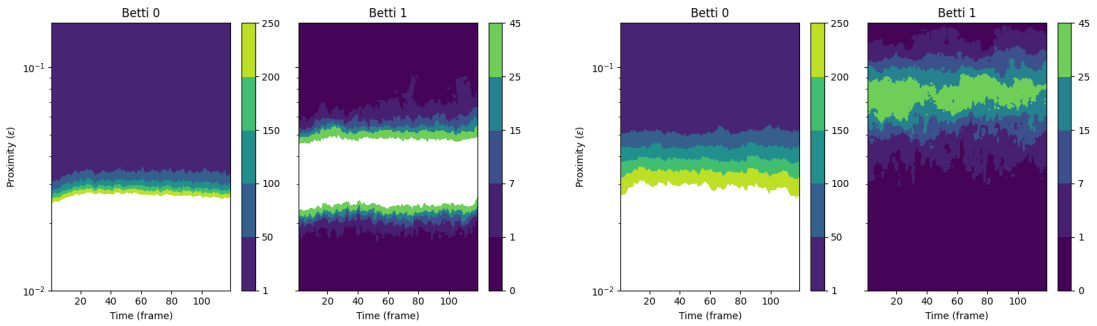

Figure 5.3: Crocker plots illustrating the contrast between the topology of a high-density dataset (left) and the same high-density dataset after trajectory subsampling (right).

in TDA is the *bottleneck distance* between persistence diagrams, which is a rigorous method for contrasting specific topological features based on their birth and death times [81]. Because Betti numbers do not keep track of individual topological features, there is no generalization of this metric for any measures based on Betti numbers, including crockers. Thus, to obtain crocker matrices for high-density scratch assay datasets, we need a method for artificially reducing the cardinality of the dataset in such a way that a high-density dataset matches a low-density dataset, while also retaining the other measured properties of the cells.

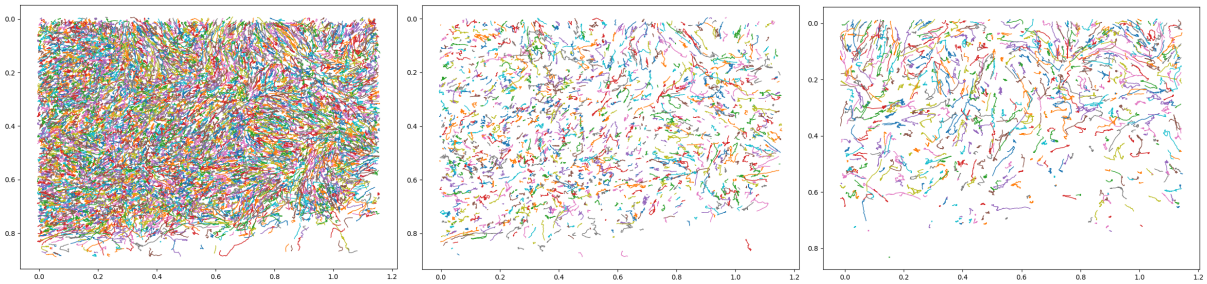

Figure 5.4: Trajectory plots of the original high-density experiment (left), the subsampled trajectories from the same experiment (center), and the corresponding low-density experiment (right). Note that, while the density is much reduced, many of the trajectories following flow channels remain.

We accomplish this by sampling trajectories from the high-density datasets to match the number of trajectories in corresponding low-density datasets. In particular, we pair a high-density trajectory dataset with an equivalent low-density dataset in terms of scratch placement, scratch shape, and general visual similarity, and then sample cells in each

frame so that the number of particles matches at each time step. We do not, however, want to blindly and randomly sample an entirely new set of particles at each frame in the high-density dataset, as this would eliminate any continuity in the dynamics. Instead, we first keep all the particles used in one frame for use in subsequent frames as long as they continue to be detected, only sampling from the remaining particles to replace ones that have left the frame or otherwise gone missing. In the rare case that we have more kept trajectories than required, we randomly cut extraneous particles from the frame and continue as before.

In this way we obtain *subsampled* high-density trajectory datasets, where each frame has the same overall density as the corresponding low-density frame. Yet the properties of the remaining high-density trajectories are left unchanged, so any distinct migration behaviors exhibited by those cells should be discernible.

5.2.2 Biological Model Simulations

We know the D'Orsogna model is not an ideal framework for modeling the migration of fibroblast cells, primarily because it ignores effects of cell morphology and variation between cells in any form. Furthermore, the idea of "attraction" between fibroblasts in a scratch assay experiment is an oversimplification of more complicated chemotactic signaling at best, and could in fact be entirely inaccurate [162]. Acknowledging these shortcomings, we will proceed with simulating particle data using a model modified to fit the scratch assay as well as possible, and will then compare the TDA summaries of this model data to experimental data, hypothesizing that the differences between them will reflect the dynamics that the model fails to capture. If so, these differences may be used in future work to develop methods for TDA-based model selection, and hopefully better models for fibroblast migration.

Boundary Conditions for the D'Orsogna Model

We begin this process by slightly modifying our boundary conditions to match the experimental setup of our scratch assay videos. We previously defined our boundary conditions as a simple "no-flux" condition, which results in particles switching direction if they pass outside the boundary. In reality, the approximately constant number of cells across frames of the scratch assay dataset is likely due to an equal number of cells leaving in any non-scratch direction as those that arrive from neighboring frames. This is, however, still consistent with the implementation of the boundary conditions if we think of these boundary cells as leaving and being replaced by another cell moving in the

opposite direction.

In addition to this conceptual change, we also remove the boundary condition from the scratch-facing side of the domain. Because each experimental data position with a scratch on either the top or the bottom of the frame has a similarly-voided area in the direction of the scratch outside the frame, allowing cells to migrate out into the scratch, this is a more reasonable use of the boundary condition. Computationally, we simply do not impose a boundary condition on the bottom edge of the frames in question, as they are contained within the scratch, and remove cells that have left the boundary of the frame from topology calculations. Thus, we use the dimensional D'Orsogna models:

$$\begin{aligned}
\frac{\partial \vec{x}_i}{\partial t} &= \begin{cases} \vec{v}_i & \vec{x}_i \in \Omega \\ -\operatorname{sgn}(\vec{v}_i)_1 \ \hat{i} \cdot \vec{v}_i & (\vec{x}_i)_1 > x_b \\ \operatorname{sgn}(\vec{v}_i)_1 \ \hat{i} \cdot \vec{v}_i & (\vec{x}_i)_1 < x_a \\ \operatorname{sgn}(\vec{v}_i)_2 \ \hat{j} \cdot \vec{v}_i & (\vec{x}_i)_2 < y_a \end{cases} \\
m\frac{\partial \vec{v}_i}{\partial t} &= \left(\alpha - \beta|\vec{v}_i|^2\right)\vec{v}_i - \nabla_i U\left(\vec{x}_i\right) \\
U\left(\vec{x}_i\right) &= \sum_{j \neq i} C_r e^{-|\vec{x}_i - \vec{x}_j|/\ell_r} - C_a e^{-|\vec{x}_i - \vec{x}_j|/\ell_a}
\end{aligned} \tag{5.1}$$

and

$$\begin{aligned}
\frac{\partial \vec{X}_i}{\partial t} &= \vec{\sigma}\ \mathrm{d}W + \begin{cases} \vec{v}_i & \vec{x}_i \in \Omega \\ -\operatorname{sgn}(\vec{v}_i)_1 \ \hat{i} \cdot \vec{v}_i & (\vec{x}_i)_1 > x_b \\ \operatorname{sgn}(\vec{v}_i)_1 \ \hat{i} \cdot \vec{v}_i & (\vec{x}_i)_1 < x_a \\ \operatorname{sgn}(\vec{v}_i)_2 \ \hat{j} \cdot \vec{v}_i & (\vec{x}_i)_2 < y_a \end{cases} \\
m\frac{\partial \vec{v}_i}{\partial t} &= \left(\alpha - \beta|\vec{v}_i|^2\right)\vec{v}_i - \nabla_i U\left(\vec{x}_i\right) \\
U\left(\vec{x}_i\right) &= \sum_{j \neq i} C_r e^{-|\vec{x}_i - \vec{x}_j|/\ell_r} - C_a e^{-|\vec{x}_i - \vec{x}_j|/\ell_a}
\end{aligned} \tag{5.2}$$

where the N identical particles are indexed by $i \in 1, 2, 3, ..., N$ and $\Omega = [x_a, x_b] \times [y_a, y_b]$ is the positional domain.

The true spatial domain is $[0.0, 1.162]\ mm \times [0.0, 0.885]\ mm$, obtained by converting the pixel dimensions of 1344-by-1024 to millimeters using the provided conversion factor of 1.157 pixels/micron. However, likely due to the discrete drift correction employed in the collection process, the experimental trajectories often have values that lie just outside these intervals. Thus, to provide a fair representation of the spatial limitations to the simulated particles, we use a positional domain of $\Omega = [-0.05, 1.212]\ mm \times [-0.05, 0.935]$.

Note that, while the velocities are theoretically unbounded in the model, in practice their maximum is set by the model parameters (see discussion below), and the same is true for the experimental data and physical bounds on velocity. We will compensate for the inexact spatial boundaries when required, like in TDA calculations.

Model Parametrization

Next, we choose dimensional parameter values that match the experimental data. These parameters are selected to reflect observable characteristics of cells in the scratch assay datasets. In many cases these are recovered from the data or from known properties of the 3T3 fibroblast cells. We may also in some cases work backwards from our nondimensionalized parameter values used in (4.8) and (4.9). The first assertion we make, similar to that of our nondimensionalized model in previous chapters, is that all cells have an identical, uniform mass. Because this mass only shows up as a linear component of other parameters, we need not define it in terms of real physical units, and instead use an arbitrary unit mass setting $m = 1\ U$. This greatly simplifies the model simulation, and leads to simpler dimensional analysis of other model parameters in the investigation of reasonable parameter values.

We begin by fixing the repulsive radius to the approximate radius of a fibroblast nucleus. The radius of the nucleus should be between 4 and 5 microns [162], so we slightly increase this to account for a minimum amount of cellular material, and set $\ell_r = 0.006\ mm$. Then, working off our nondimensional parameters and citing the "biological relevance" criterion of the D'Orsogna model follow-up paper [116], we choose $\ell_a = 0.012\ mm$ so that $\ell = \ell_r/\ell_a = 1/2$. We follow a similar process to set C_a based on a desired ratio of $C = C_r/C_a = 2$, but it is more difficult to choose C_r as its units of energy are less directly measurable. We start by setting $C_r = 0.004\ U \cdot mm^2/hr^2$ where U is the unknown unit of the uniform mass of a cell as above. This parameter value has the correct order of magnitude, and is chosen so that the simulated particles avoid unrealistic clumping behavior. Using the predetermined ratio of $C = 2$, we obtain $C_a = 0.002\ U \cdot mm^2/hr^2$. Lastly, we set α and β so that the distribution of velocities are approximately equal between the datasets while still maintaining desired behavior, as the D'Orsogna model papers show that the ratio between the two determines the equilibrium velocity of the particles. We first set $\beta = 0.1\ U \cdot hr/mm^2$, and then attempt $\alpha \approx 0.000459\ U/hr$ so that $\sqrt{\alpha/\beta} = 0.06773\ mm/hr$, the maximum observed particle speed. However this results in the distinct congealing behavior which does not occur in our experimental data. Since this is caused by insufficient self-propulsion with respect to the attractive forces, we increase the relative strength of the self-propulsion by setting $\alpha = 0.01\ U/hr$. Although this

results in higher maximum simulated velocities, this stronger self-propulsion eliminates clumping and brings about a reasonably similar distribution of velocities over the course of the entire experiment, as shown in Figure 5.5.

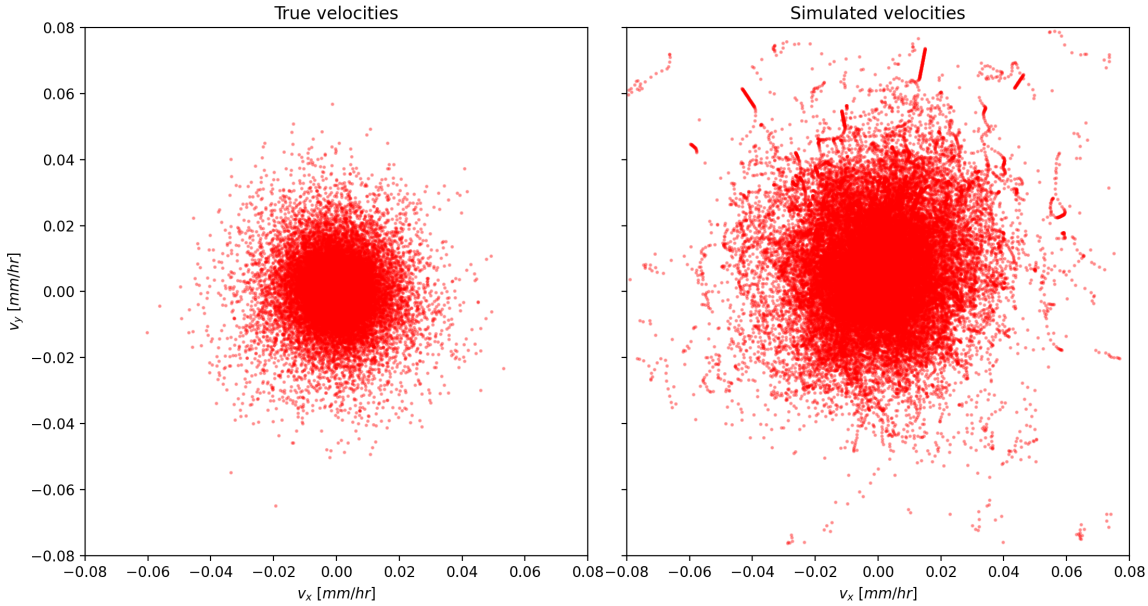

Figure 5.5: Scatter plots of the velocities of all particles in all frames from both the true dataset (right) and the simulated dataset (left).

Initial Conditions

Finally, following the example of the aforementioned model selection experiment [98], we use real cell positions and velocities as the initial condition for the model. Due to excessive observational noise in early frames, we use the particles from a single frame collected 1 hour into the experiment as the initial condition. We sample subsequent positions and velocities at the same time scale as the experimental data, with 10 minutes or 1/6 of an hour between frames. These datasets are simulated until the end time of the true data. Thus each dataset consists of 121 frames; one for each 10 minute interval between 1 hour and 21 1/6 hours.

Our simulated datasets are seeded using positions and velocities from the 337 cells detected in the 7th frame of the video collected at position 9, a low-density position in the experimental dataset. This results in 121 separate clouds of 337 points in 4-dimensional

space. The deterministic model produces identical results when seeded with the same initial conditions, so we need only one deterministic simulation. Likewise, we create 10 datasets from the stochastic model using the same model parameters, initial conditions, and time points in order to have a representative sample of potential model output datasets.

5.2.3 Crocker Plot Hyperparameters

The mathematical details of the crocker operator are introduced in Chapter 3 and thoroughly discussed in Chapter 4. The computation we use here is similar to what we use in our parameter estimation experiments. We maintain the use of exponentially-spaced proximity parameter values and use the same procedure of acquiring Betti numbers from the filtration of the Vietoris-Rips complex. Primary differences include using more time points, using more proximity values, and computing the crocker matrices for both the 0th Betti number (b_0) and the 1st Betti number (b_1).

Specifically, we compute the crockers for each of our trajectory datasets in question using 400 exponentially-spaced proximity values. For the crockers made using the entire 4-dimensional point cloud we use values ε_j spaced exponentially with base 10 between 10^{-2} and $10^{-0.8}$:

$$\varepsilon_j = 10^{\left[-2+(j-1)\frac{-0.8+2}{399}\right]},\ j = 1, ..., 400. \tag{5.3}$$

We also will compute crockers using only the cell velocities, for which we will use smaller proximity parameter values between 10^{-4} and $10^{-1.8}$:

$$\varepsilon_j = 10^{\left[-4+(j-1)\frac{-1.8+4}{399}\right]},\ j = 1, ..., 400. \tag{5.4}$$

We also use nearly all of the available point clouds in the time series. The first 6 frames of the real datasets cannot be used for comparison since the simulations are seeded using positions from the 7th frame, so they are discarded. The final two time points provide less reliable approximations of the cell velocities due to finite difference formulas and experimental limitations, so they are discarded. Finally, the 91st frame of our low-density dataset is missing, so we remove that frame from the subsampled high-density and simulated datasets. Thus we use all time points between 7 and 126 except for 91, a total of 119 time step point clouds per dataset. Therefore we can define the time of a point cloud t_i as

$$t_i = 1 + \frac{i-1}{6},\ i \in [1..84] \cup [86..120]. \tag{5.5}$$

Furthermore, we compute both the b_0 and b_1 crockers. To recapitulate, the 0th Betti number is a count of the connected components in a simplicial complex, and the 1st Betti number is a count of the "loops", or 1-dimensional holes. Since we are aiming for interpretability, we want to examine the changes in both of these measures. Using the understanding of Betti numbers conferred in prior research, we know that the scale of connected component births and deaths is highly dependent on agent spreading [98] for b_0, and that peaks in b_1 can be caused by sufficiently-circular clusters of cells [118].

Finally, we display the level sets for $b_0 = 1, 50, 100, 150, 200, 250, \infty$ and $b_1 = 0, 1, 7, 15, 25, 45, \infty$ when plotting the crockers, as these contours sufficiently illustrate the change in topology over the range of proximity values. Altogether, we compute two crocker matrices, $\mathbf{C}_0, \mathbf{C}_1 \in \mathbb{Z}^{400\times 119}$ for each dataset. We compare these crockers visually using the crocker plot contour method [97], drawing conclusions based on known interpretations of topology.

5.3 Results

We first examine the crocker plots for the experimental data to discern differences in the topology and further ensure the efficacy of our subsampling approach. We then compare the experimental crockers to the simulated crockers to examine how the decidedly simplified dynamics of the D'Orsogna model cause the topology to change.

5.3.1 Subsampled True Data

Full Crocker Plots

The low-density dataset (# 9) and subsampled version of the corresponding high-density dataset (# 31) have exactly the same number of particles per frame, which should serve to correct for any effect the density has on the topology. This subsampling methodology certainly seems to have worked appropriately, as the resultant crocker plots, shown in Figure 5.6 are particularly visually similar.

One of the key features is the relatively constant behavior of the 0th Betti numbers over time. In fact, very little can be gleaned from the b_0 crockers, as they are exceptionally flat and similar to each other. The only discernible difference in pattern for these is the slight increase in contour curves over the first 20 frames of the experiment in the subsampled crocker (bottom left), which are absent from the low-density crocker (top left). According to Ulmer et al. this increase suggests a dispersive behavior [98]. This is

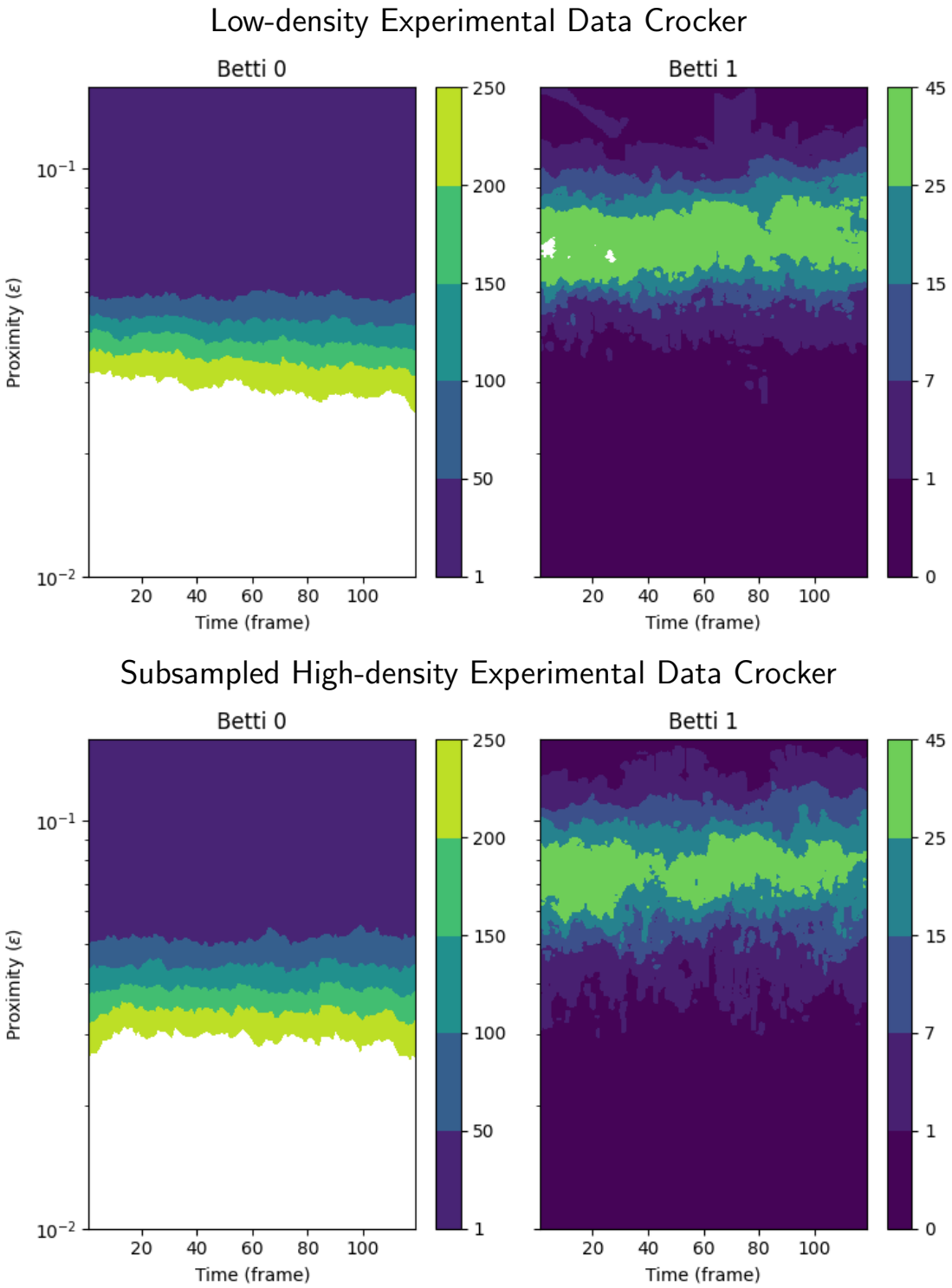

Figure 5.6: The crocker plots for the low-density experimental dataset at position 9 (top) and the subsampled high-density experimental dataset at position 31 (bottom).

consistent with the hypothesis that the high-density experiments spread more quickly into the scratch.

The 1st Betti numbers vary more than the 0th Betti numbers, but are still rather similar in the context of the high-density comparison (Figure 5.3). We can see that the low-density crocker does achieve higher b_1 numbers consistently over the same proximity parameter values and time points. There are multiple explanations for this, but one reason could be a higher count of circular clumps of cells, as described by Bonilla et al. [118]. This is again consistent with our biological hypothesis that over-confluent populations experience fluidization, resulting in fewer stationary clusters of cells. They are otherwise very similar, as the amount of variance between frames and across Betti curves is relatively consistent.

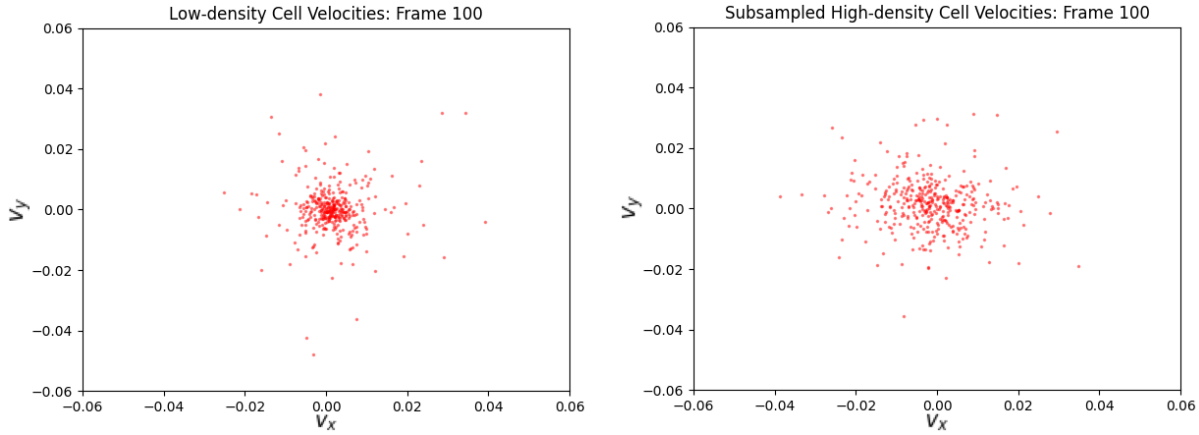

Figure 5.7: Scatter plots of cell velocities from corresponding frames in the low-density (left) and subsampled high-density (right) experimental datasets.

Velocity-Only Crocker Plots

However, if we once again consider the confounding factor that density has on topology, we would expect to see some similar patterns in crockers involving particle position. Because we have exactly the same number of cells in the same spatial domains, the topology of the position dimensions of the point cloud is relatively limited. We can account for this and attempt to uncover finer topological details by instead computing the crocker using only the velocities. In other words, we compute the VR filtration of a 2-dimensional point cloud of velocities where each point is of the form (v_x, v_y). Examples of these point clouds for a single frame of our data are shown in Figure 5.7. When compared to the above crockers

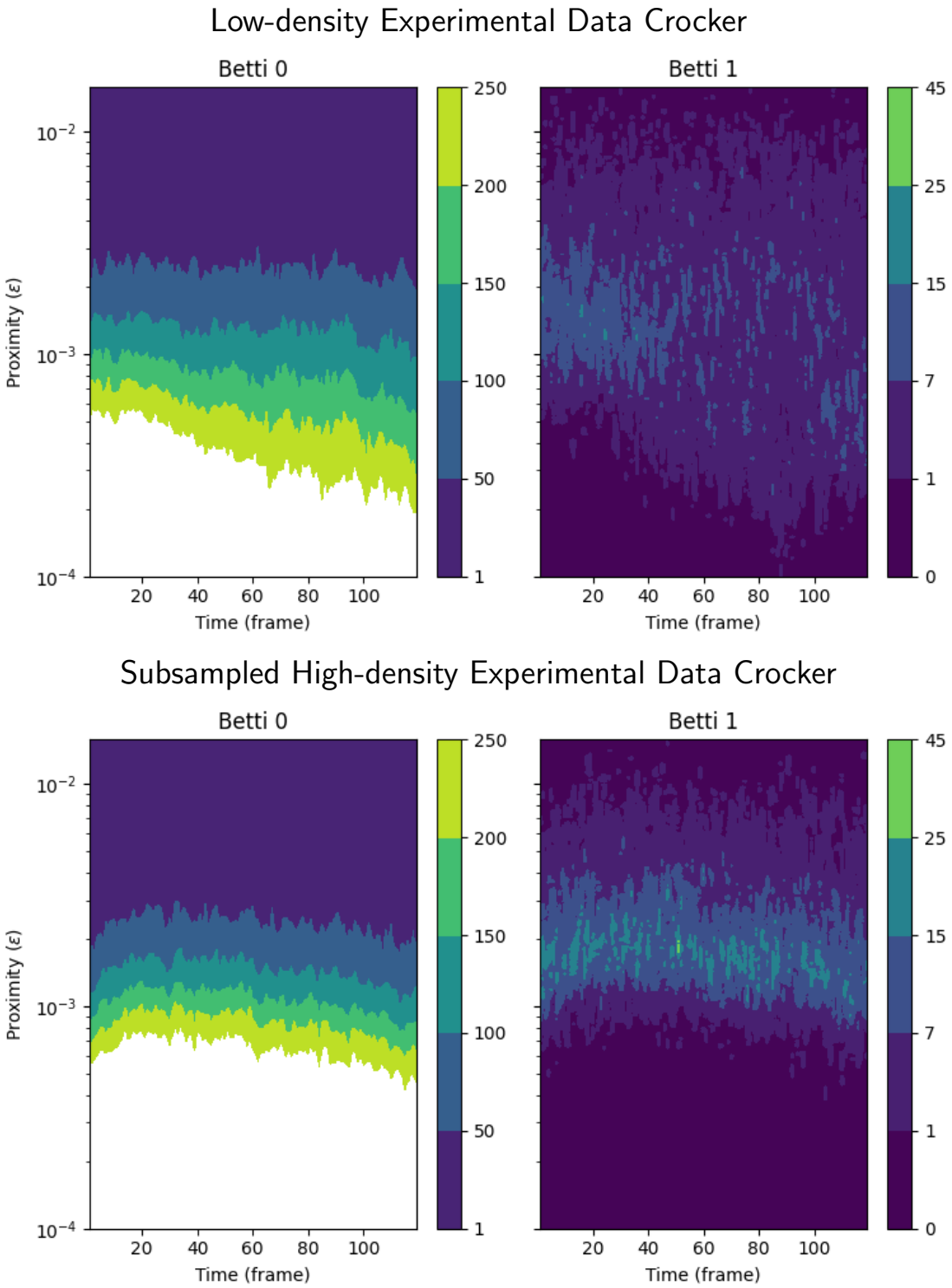

Figure 5.8: The velocity-only crocker plots for the low-density experimental dataset (top) and the subsampled high-density experimental dataset (bottom).

based on the filtrations of 4-dimensional point clouds, we will see different features as the topology of the smaller-scale velocities are not overshadowed by the positions. Figure 5.8 shows the velocity-only crocker plots corresponding to the crockers in Figure 5.6.

These crocker plots highlight the distinctly different behaviors, but are less immediately interpretable. Just as the topology of the 4-dimensional point cloud was dominated by the cell positions in the previous experiment, so was our evaluation influenced most by our understanding of positional topological features. Thus we must reframe the significance of topological features, given we are now examining the topology of the velocity space. The decreasing pattern in the b_0 can be compared to the increasing pattern in the same Betti number of the full crocker plots. Similar to how the increasing pattern signaled a spreading of cell positions, the decrease indicates the contraction of the velocity space. Physically, this likely means the low-density cells move more slowly and fewer cells move at all once the population has spread into the scratch. In contrast, the subsampled high-density velocities show a flatter b_0 behavior over time, suggesting they continue with approximately the same distribution of velocities.

The crocker plots for the 1st Betti numbers are, as with the full crockers, more varied and less obviously correlated with visible patterns. However, a notable distinction between these b_1 crockers and those produced by the 4-dimensional point clouds is that they follow the behavior of the b_0 at the same proximity parameter values. This pattern is especially pronounced, where the increase and subsequent decrease over time clearly illustrated in the Betti 0 crocker is also present in the contour curves of the Betti 1 plot. The subsampled high-density Betti 1 crocker also has consistently higher values. Just as this behavior indicates a higher number of sufficiently-round clumps when applied to positional data, this may also be interpreted as clumps of similar velocity values. This could be caused by a number of biological behaviors. For example, it could be the result of channels of multiple faster moving cells, a pattern that often arises in the aforementioned cell fluidization paradigm.

While some of these behaviors seem readily apparent, many of them are not revealed by other population summary investigations. For example, the mean particle speed or maximum particle velocity is not generally a good indicator of whether or not an experimental dataset is low-density or high-density as shown in Figure 5.9. As suggested by the scatter plot (Figure 5.7), there may be a stronger correlation between the velocity variance and experimental cell density. We examine this by plotting the sample variances of the velocity components over time in Figure 5.10. This trend is confirmed only for the x-component of the velocities, as the y-component is very weakly correlated in later frames. This is likely due to the movement in the y direction resulting primarily from

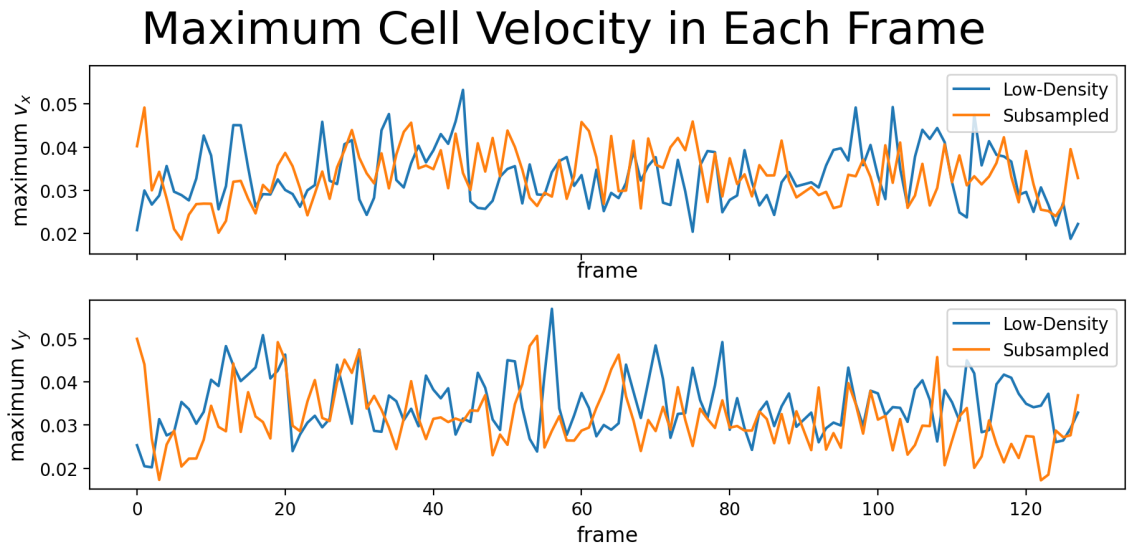

Figure 5.9: Plots of the maximum component velocities of the low-density and subsampled high-density datasets over time.

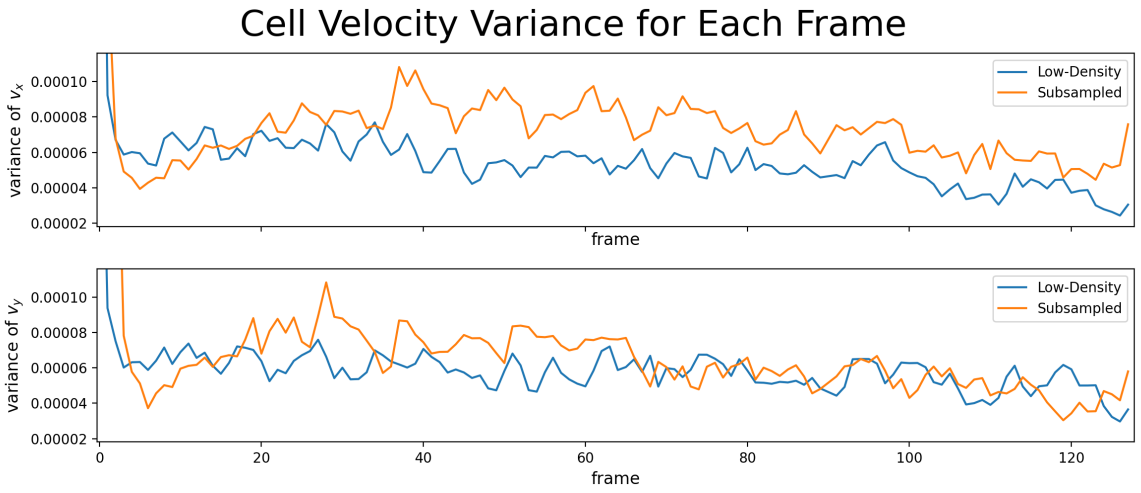

Figure 5.10: Plots of the variance of the velocity components of the low-density and subsampled high-density datasets over time.

migration into the scratch. While this shows that other simple evaluations of the velocity can show distinctions between the datasets, these scalar values offer little in terms of interpretability, and thus do not reveal the nuanced biological behaviors we would like to understand. These findings indicate that topological summary methods are useful for distinguishing the point cloud characteristics that result from these differing biological factors such as population density.

5.3.2 Simulated Data

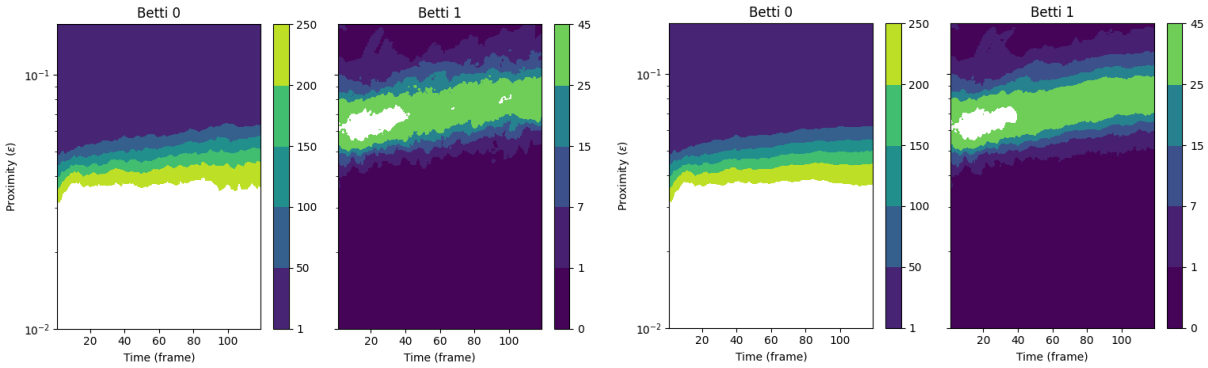

Figure 5.11: Crocker plots for the Betti numbers of a single stochastic model simulation (left) and the average Betti numbers over all 10 stochastic simulations (right).

We may now apply the same crocker plot evaluation to our simulated data as a means to contrast it with our experimental data. While we have already seen ways in which the simulations differ from the true data (see Figure 5.5), applying comparative TDA to these data will illustrate the ways these differences impact the topology of the underlying point clouds, and help us to further investigate the explainability of TDA methods as applied to real biological data. We first justify the use of a single stochastic simulation crocker plot by using the methodology employed by Ulmer et al. and comparing the crocker plot consisting of Betti numbers averaged element-wise over the 10 realizations with a single stochastic plot. Figure 5.11 shows both the mean stochastic crocker plot and a representative plot for a single simulation. It is immediately evident that the level of noise we use in our simulations does not result in significant changes in the general patterns of the topology, so averaging them appears to only "smooth out" the contours of the crocker plot. For this reason, we choose to draw comparisons between a single simulation crocker plot and the low-density crocker plot whose positions seeded the simulation, as shown in Figure 5.12.

Certain patterns in the crocker plot from this single simulation are easily explained by causes that we have already explored. The first few time points show the sharp increase in Betti numbers characteristic of spreading particles [98]. This behavior is easily explained by the higher cell velocities in the simulation compared to the experimental data. The higher b_1 values at similar proximity parameter values can also be explained by this rapid spreading, as this results in more clumps of cells that are set apart from

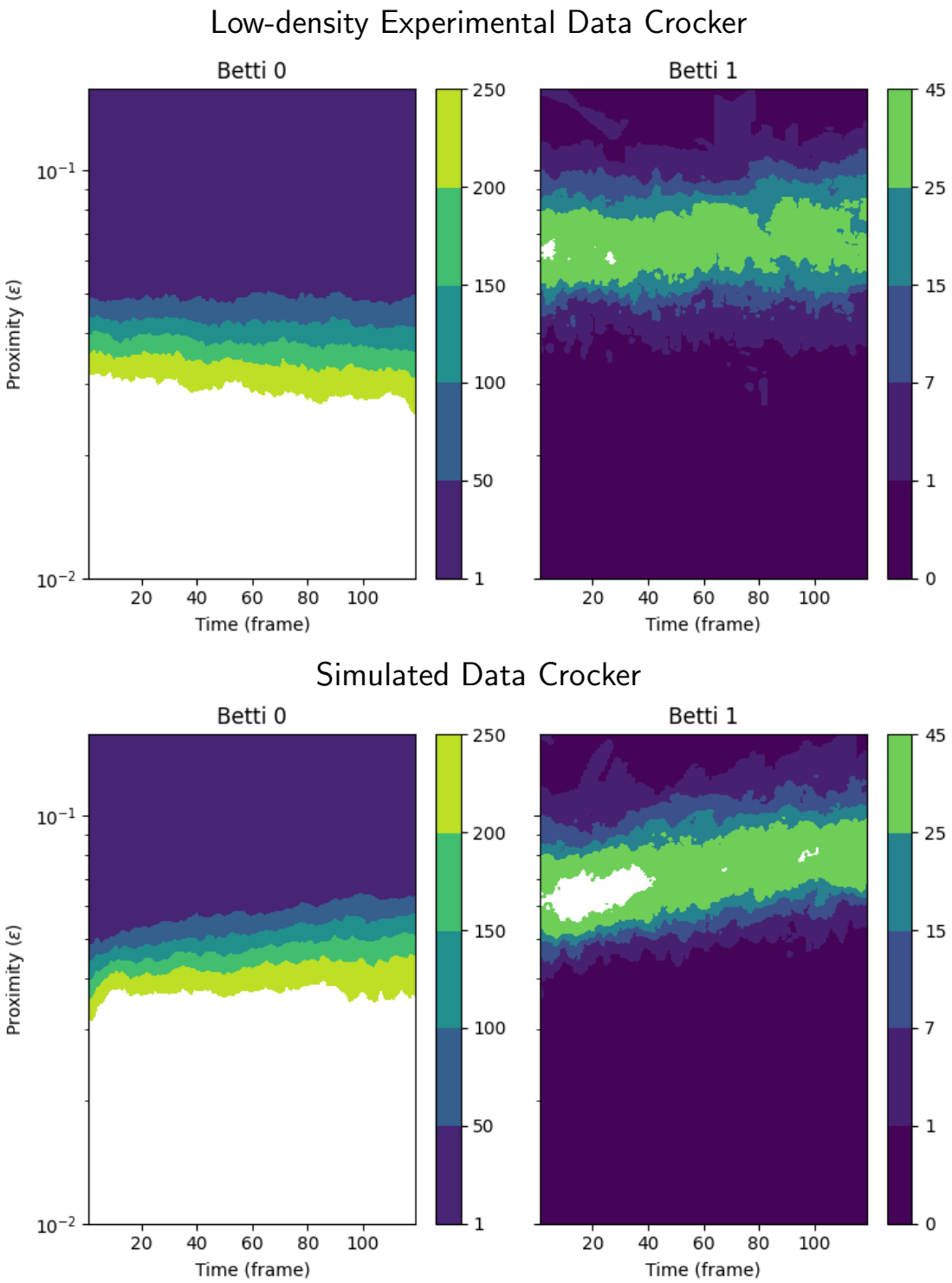

Figure 5.12: The crocker plots for the low-density experimental dataset (top) and a single stochastic simulated dataset (bottom).

the more densely-spaced primary population. There are, however, other features that are encouragingly similar to the true experimental data, primarily related to the emergent behaviors of the cells. The most notable of these is the fluctuation of the b_1 Betti curves for individual time points. For example, we see a few instances the 1st Betti number decreasing and then increasing again as the proximity parameter increases. These appear as "islands" of higher valued contours, and are particularly visible in the lower contours around frames 40 and 90 in addition to being prevalent in the higher contours. While there are more cases of this pattern in the experimental data than in the simulated data, this quality is an indicator of high local *polarization*—cells moving in different directions in close proximity—as shown by Bhaskar and colleagues [117]. These fluctuations are rarely seen in particle populations with homogeneous motion, epitomized by the single mill phenotype introduced in preceding chapters.

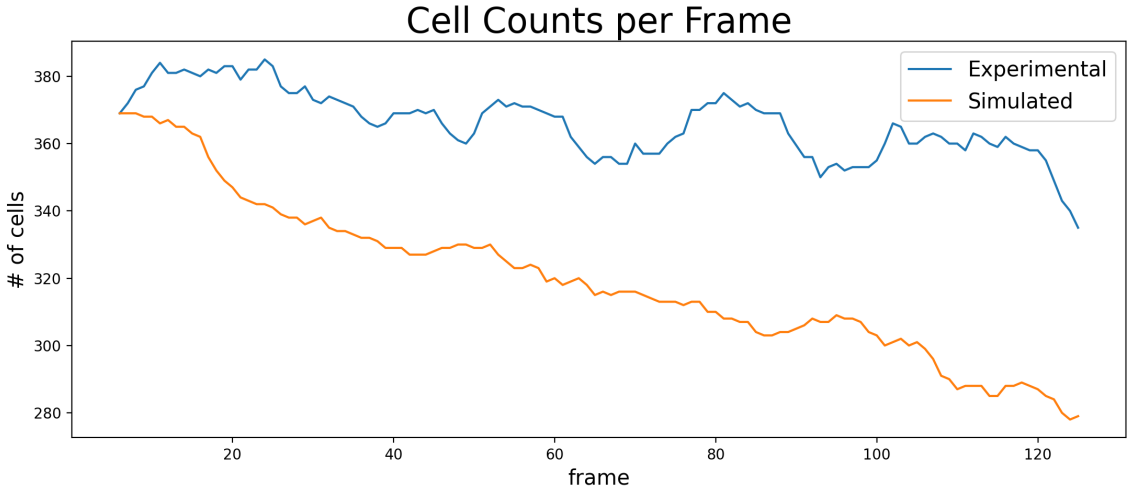

Figure 5.13: A plot of the number of particles in each frame of the low-density experimental and stochastic simulated datasets.

The velocity-only crocker plot demonstrates many of the same patterns discussed above, and on first examination does not reveal any further insights into the differences between simulated and experimental data. Thus, rather than discussing the velocity-only crocker for simulation data, we will again discuss the perplexing effect of changes in density on topology. We first examine the difference in density between the experimental and simulated datasets by plotting the number of cells within the frame boundary over time (Figure 5.13). While the simulation is seeded with exactly the same cells as those in the corresponding experimental frame, the divergent locomotive behavior results in

substantially different dynamics. Subsequently, more cells are lost through the scratch in the simulated dataset. The difference in cell count is further exacerbated by the apparent increase in accurately detected cells in the experimental dataset, and the lack of cell division in the model. While the number of cells does decrease in both datasets, by the end of the time span the simulated point cloud has only about 83% as many particles as the experimental point cloud. Our TDA measure reflects this difference in point cloud size, further proving its ability to capture multi-scale dynamics.

5.4 Contributions

Our crocker plot investigation demonstrates the distinct differences between the data produced by these simple particle motion models and the true experimental data. We can recover some of the unique cell migration dynamics that are not accurately modeled in the D'Orsogna model variants, but which are present in the interactions of dense monolayers of fibroblasts, by examining the topology of the locations and velocities of these cells. Even though we model cells as dimensionless particles, we can expose behaviors that are characteristic of morphologically-driven cell interactions. Because we can discern some of these features in relatively simple TDA visualizations with a trained eye, we are confident that these summaries of topology will be useful for rigorous algorithmic approaches.

A surprising finding is the relative uniformity of the stochastic crocker plots. Despite a significant amount of noise which—as shown in Chapter 4—propagates nonlinearly through time in the simulated data, these crocker plot visualizations demonstrate nearly identical topology between stochastic realizations, illustrated in Figure 5.11. This is particularly encouraging in relation to the evaluation of biological datasets, since they are inherently noisy. Because crocker plots reveal global patterns in the midst of randomness, they demonstrate the stability needed to assess large-scale behaviors. However, by being sufficiently sensitive to specific dynamics, they are also useful as a quantitative measure of dataset comparison.

Generalizing this idea, we hypothesize that TDA is sufficiently sensitive to multi-scale cell migration dynamics, and may be used as a method for evaluating the suitability of more sophisticated fibroblast migration models. While the D'Orsogna model is never assumed to be an appropriate descriptor of our experimental data, we are able to determine which dynamics it more accurately produces by leveraging the power of TDA visualizations. At the same time, TDA is more robust to noise than many other summaries of collective motion behavior, such as maximum speed or the sample variance of particle velocities. Thus we believe that, combined with a concrete statistical approach for accepting and

rejecting models, TDA methods such as these may in future work be used as quantitative information for the development of uniquely-broad model selection criteria. Our findings on the application of crocker plots to complex cell migration data suggest that this approach could be used to evaluate advanced multi-scale agent-based cell migration models.

Chapter 6

Conclusions

This chapter briefly summarizes the objectives of this dissertation and suggests avenues of continued investigation.

6.1 Summary of Work

The primary undertaking of this research is to better understand collective motion from a mathematical and data scientific perspective. We accomplish this with three distinct but interrelated projects. Firstly, we use an original deep convolutional neural network approach to acquire accurate point locations of cell nuclei in microscopy images. Any mathematical investigation requires quantitative data, and by combining this computer vision technique with a pragmatic particle linking algorithm, we obtain accurate collective motion point clouds. Second, we apply topological data analysis to the task of estimating stochastic collective motion model parameters. By leveraging novel topological tools, we solve inverse problems involving infinitesimal parameter value increments, demonstrating the ability of TDA to discern small-scale particle interactions among randomness and nonlinearity. Finally, we investigate the capabilities of TDA in the form of crocker plots, particularly as a method for evaluating collective motion in both experimental data and agent-based model simulation. We demonstrate the robustness of crocker plots to non-normal, multi-dimensional random noise, and the sensitivity of crockers to large-scale emergent behaviors, both of which suggest TDA as a powerful means for selecting models based on specific collective motion characteristics. Altogether, we further the understanding of collective motion by demonstrating both an ability to obtain meaningful information from unseen experimental data, and the functionality of crocker matrices for analyzing meaningful and often-interpretable patterns on a variety of scales.

Data science is primarily motivated by the abundance of data available to modern-day

researchers. Yet many interesting questions require new and innovative investigations and the collection of novel data. Our cell tracking results show that we may still analyze these new datasets with state-of-the-art data science methods, despite a lack of labeled training data. We are able to turn completely unprocessed imagery into sizable point cloud datasets whose shape, size, and behavior can be evaluated mathematically. Topological data analysis is based on rigorous ideas of shape and geometric connectivity that guarantee stability and interpretability. By applying TDA to complex, noisy datasets, we demonstrate the benefits of this robustness. Simple matrix summaries of topology in the form of crockers are at once able to capture both precise discrepancies between high-dimensional, nonlinear candidate models for the sake of parameter estimation, and global phenotypes that distinguish cell migration patterns, arising from distinctly different populations. When combining these approaches, we gain insight into the vast capabilities of data-driven mathematical investigation. We apply these methods to concrete problems in collective motion research and, as a result, uncover meaning from unexplored data. This allows us to both quantify the expected behaviors and hypothesized results, and discover entirely new relationships between small and large scales, discrete and continuous measurements, and heterogeneous and collective behaviors.

6.2 Future Work

Ongoing research involving topological data analysis and the assessment of collective motion behaviors can focus on the varied scales of use for TDA as it relates to the evaluation of candidate cell migration models. Having shown the capacity of crockers to capture meaning at multiple scales and provide meaningful qualitative and quantitative contrasts between distinct behaviors, we recommend leveraging topology for end-to-end model selection and parameter estimation of agent-based models. Because crocker plots provide clear visualization of global behaviors and are sufficiently robust to noise, they are well-suited for direct model comparison. Researchers will be able to select cell migration models based on their ability to express desired population behaviors, such as contact inhibition of locomotion or cellular fluidization. Furthermore, since vectorized crocker matrices are also adequately sensitive to subtle changes in populations over time, they can be used to evaluate the heterogeneous behaviors that distinguish noisy data from truth. Overall, data-driven topological analysis of mathematical collective motion models shows promise for the discovery of important new behaviors, and the solution of open questions in cell migration. We anticipate the application of TDA to agent-based model candidates that incorporate important biological factors, including accurate interactions

between cells due to chemotaxis and contact inhibition of locomotion, the impacts of environmental factors, and the effects of cell morphology.

We have furthermore laid the groundwork for more rigorous investigation of parameter estimation involving topological forward problems. In particular, by using a probabilistic inverse problem framework, we enable the extension of our estimation methodology into Bayesian analysis. A more accurate statistical model allows the use of likelihood functions for Bayesian parameter estimation, which not only guarantees certain properties for the resulting the posterior parameter distribution, but also facilitates the quantification of uncertainty. Bayesian uncertainty quantification provides a framework for direct comparison of parameter estimation methods, in this case revealing the relative merits of different approaches to summarizing model behavior.

BIBLIOGRAPHY

[1] Blei, D. M. & Smyth, P. "Science and data science". *Proceedings of the National Academy of Sciences* **114**.33 (2017), pp. 8689–8692. eprint: `https://www.pnas.org/content/114/33/8689.full.pdf`.

[2] Goodfellow, I. et al. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016.

[3] Weisstein, E. W. *Optimization Theory*. 2021. URL: `https://mathworld.wolfram.com/OptimizationTheory.html` (visited on 03/12/2021).

[4] Marshall, C. *The Magic Roundabout —Roads.org.uk*. 2021. URL: `https://www.roads.org.uk/articles/the-magic-roundabout` (visited on 03/13/2021).

[5] Reynolds, C. W. "Flocks, Herds and Schools: A Distributed Behavioral Model". *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, pp. 25–34.

[6] Vicsek, T. & Zafeiris, A. "Collective motion". *Physics Reports* **517**.3 (2012), pp. 71–140.

[7] Stanley, H. *Introduction to Phase Transitions and Critical Phenomena*. International series of monographs on physics. Oxford University Press, 1987.

[8] Vicsek, T. et al. "Application of statistical mechanics to collective motion in biology". *Physica A: Statistical Mechanics and its Applications* **274**.1 (1999), pp. 182–189.

[9] Goldstein, J. "Emergence as a Construct: History and Issues". *Emergence* **1**.1 (1999), pp. 49–72.

[10] Bonabeau, E. "Agent-based modeling: Methods and techniques for simulating human systems". *Proceedings of the National Academy of Sciences* **99**.suppl 3 (2002), pp. 7280–7287. eprint: `https://www.pnas.org/content/99/suppl_3/7280.full.pdf`.

[11] Vicsek, T. et al. "Novel Type of Phase Transition in a System of Self-Driven Particles". *Phys. Rev. Lett.* **75** (6 1995), pp. 1226–1229.

[12] Scott, T. *The Magic Roundabout: Swindon's Terrifying Traffic Circle and Emergent Behaviour*. 12, 2015. URL: `https://youtu.be/D22BOOGbpFM` (visited on 03/13/2021).

[13] Lai, W. et al. *Introduction to Continuum Mechanics*. Elsevier Science, 2014.

[14] Metzcar, J. et al. “A Review of Cell-Based Computational Modeling in Cancer Biology”. *JCO Clinical Cancer Informatics* **3**.3 (2019). PMID: 30715927, pp. 1–13.

[15] Matsiaka, O. M. et al. “Mechanistic and experimental models of cell migration reveal the importance of cell-to-cell pushing in cell invasion”. *Biomedical Physics & Engineering Express* **5**.4 (2019), p. 045009.

[16] González-Valverde, I. & García-Aznar, J. M. “Mechanical modeling of collective cell migration: An agent-based and continuum material approach”. *Computer Methods in Applied Mechanics and Engineering* **337** (2018), pp. 246–262.

[17] Baker, R. E. & Simpson, M. J. “Models of collective cell motion for cell populations with different aspect ratio: Diffusion, proliferation and travelling waves”. *Physica A: Statistical Mechanics and its Applications* **391**.14 (2012), pp. 3729–3750.

[18] Tukey, J. W. *Exploratory data analysis*. Reading, Mass.: Addison-Wesley Pub. Co., 1977.

[19] Weisstein, E. W. *Hilbert Space*. 2021. URL: `https://mathworld.wolfram.com/HilbertSpace.html` (visited on 02/20/2021).

[20] Chang, H. H. et al. “Transcriptome-wide noise controls lineage choice in mammalian progenitor cells”. *Nature* **453**.7194 (2008), pp. 544–547.

[21] Kwon, T. et al. “Stochastic and Heterogeneous Cancer Cell Migration: Experiment and Theory”. *Scientific Reports* **9**.1 (2019), p. 16297.

[22] Lomvardas, S. et al. “Interchromosomal Interactions and Olfactory Receptor Choice”. *Cell* **126**.2 (2006), pp. 403–413.

[23] Altschuler, S. J. & Wu, L. F. “Cellular Heterogeneity: Do Differences Make a Difference?” *Cell* **141**.4 (14, 2010), pp. 559–563.

[24] *Participants – Cell Tracking Challenge*. Cell Tracking Challenge. 2020. URL: `http://celltrackingchallenge.net/participants/` (visited on 12/07/2020).

[25] *Single Cell Tracking —HoloMonitor Imaging Assays*. Phase Holographic Imaging PHI. 2021. URL: `https://phiab.com/applications/single-cell-tracking/` (visited on 01/20/2021).

[26] *Imaris for Cell Biologists - Imaris - Oxford Instruments*. Oxford Instruments. 2021. URL: `https://imaris.oxinst.com/products/imaris-for-cell-biologists` (visited on 01/20/2021).

[27] Löffler, K. & Scherr, T. *KIT-Sch-GE*. Cell Tracking Challenge. 2020. URL: `https://public.celltrackingchallenge.net/participants/KIT-Sch-GE.pdf` (visited on 02/19/2021).

[28] Lux, F. & Matula, P. *MU-Lux-CZ*. Cell Tracking Challenge. 2020. URL: https://public.celltrackingchallenge.net/participants/MU-Lux-CZ.pdf (visited on 02/19/2021).

[29] Ren, S. et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". *Advances in Neural Information Processing Systems*. Ed. by Cortes, C. et al. Vol. 28. Curran Associates, Inc., 2015, pp. 91–99.

[30] Redmon, J. et al. "You Only Look Once: Unified, Real-Time Object Detection". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[31] Chanda, S. et al. "Face Recognition - A One-Shot Learning Perspective". *2019 15th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*. Vol. 15. 2019, pp. 113–119.

[32] Hsieh, T.-I. et al. "One-Shot Object Detection with Co-Attention and Co-Excitation". *Advances in Neural Information Processing Systems*. Ed. by Wallach, H. et al. Vol. 32. Curran Associates, Inc., 2019.

[33] Crocker, J. C. & Grier, D. G. "Methods of Digital Video Microscopy for Colloidal Studies". *Journal of Colloid and Interface Science* **179**.1 (1996). Reprinted with permission from Elsevier, pp. 298–310.

[34] *NIH 3T3 Cell Line Origins, Characteristics, Transfection - NIH 3T3 Cell Line*. Altogen Biosystems. 2021. URL: nih3t3.com (visited on 01/19/2021).

[35] Bear, J. E. & Haugh, J. M. "Directed migration of mesenchymal cells: where signaling and the cytoskeleton meet". *Current opinion in cell biology* **30** (2014), pp. 74–82.

[36] Mayor, R. & Carmona-Fontaine, C. "Keeping in touch with contact inhibition of locomotion". *Trends in Cell Biology* **20**.6 (1, 2010), pp. 319–328.

[37] Sen, C. K. "Human Wounds and Its Burden: An Updated Compendium of Estimates". *Advances in Wound Care* **8** (2 14, 2019), pp. 39–48.

[38] Dees, C. et al. "Cellular and molecular mechanisms in fibrosis". *Experimental dermatology* **40**.1 (2021), pp. 121–131.

[39] Passucci, G. et al. "Identifying the mechanism for superdiffusivity in mouse fibroblast motility". *PLOS Computational Biology* **15**.2 (2019), pp. 1–15.

[40] *Product information: SiR-DNA (SC007)*. 2018. URL: https://www.cytoskeleton.com/pdf-storage/datasheets/cy-sc007.pdf (visited on 03/06/2021).

[41] Dictionary, M.-W. *Confluence*. Entry 3. Merriam-Webster, 2021.

[42] Liang, C.-C. et al. "In vitro scratch assay: a convenient and inexpensive method for analysis of cell migration in vitro". *Nature Protocols* **2**.2 (1, 2007), pp. 329–333.

[43] Trackpy Contributors. *Trackpy: Fast, Flexible Particle-Tracking Toolkit —trackpy 0.4.2 documentation*. 2019. URL: `http://soft-matter.github.io/trackpy/v0.4.2/index.html` (visited on 03/06/2021).

[44] Trackpy Contributors. *Introduction to Trackpy — trackpy 0.4.2 documentation*. 2019. URL: `http://soft-matter.github.io/trackpy/v0.4.2/introduction.html` (visited on 03/06/2021).

[45] Ronneberger, O. et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation". *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Navab, N. et al. Springer International Publishing, 2015, pp. 234–241.

[46] Hastie, T. et al. *The Elements of Statistical Learning*. 2nd ed. New York, NY: Springer-Verlag, 2009, pp. 392–399.

[47] Hastie, T. et al. *The Elements of Statistical Learning*. 2nd ed. New York, NY: Springer-Verlag, 2009, pp. 119–120.

[48] *tf.keras.losses.CategoricalCrossentropy —TensorFlow Core v2.4.1*. 2021. URL: `https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy` (visited on 02/18/2021).

[49] Ketkar, N. *Stochastic Gradient Descent. Deep Learning with Python: A Hands-on Introduction*. Berkeley, CA: Apress, 2017, pp. 113–132.

[50] Orr, G. *Momentum and Learning Rate Adaptation*. Department of Computer Science, Willamette University. URL: `https://www.willamette.edu/~gorr/classes/cs449/momrate.html` (visited on 03/07/2021).

[51] Goodfellow, I. et al. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016, pp. 296–300.

[52] Höfener, H. et al. "Deep learning nuclei detection: A simple approach can deliver state-of-the-art results". *Computerized Medical Imaging and Graphics* **70** (2018), pp. 43–52.

[53] Long, J. et al. "Fully convolutional networks for semantic segmentation". *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

[54] Tompson, J. et al. "Efficient object localization using Convolutional Networks". *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 648–656.

[55] Chenouard, N. et al. "Objective comparison of particle tracking methods". *Nature Methods* **11**.3 (2014), pp. 281–289.

[56] Crocker, J. C. & Weeks, E. R. *Explanation of tracking macros*. 24, 2006. URL: http://www.physics.emory.edu/faculty/weeks//idl/tracking.html (visited on 03/06/2021).

[57] Trackpy Contributors. *trackpy/center_of_mass.py*. 2019. URL: https://github.com/soft-matter/trackpy/blob/d8cd840f594db5478ed6210d8ed4a6dbceef09cc/trackpy/refine/center_of_mass.py (visited on 03/06/2021).

[58] Trackpy Contributors. *walkthrough — trackpy 0.4.2 documentation*. 2019. URL: http://soft-matter.github.io/trackpy/v0.4.2/tutorial/walkthrough.html (visited on 03/06/2021).

[59] Trackpy Contributors. *Trackpy: Fast, Flexible Particle-Tracking Toolkit —trackpy 0.4.2 documentation*. 2019. URL: http://soft-matter.github.io/trackpy/v0.4.2/generated/trackpy.link.html (visited on 03/06/2021).

[60] Allan, D. et al. *soft-matter/trackpy: Trackpy v0.4.2*. Version v0.4.2. 2019.

[61] Reback, J. et al. *pandas-dev/pandas: Pandas 1.1.0*. Version v1.1.0. 2020.

[62] *scipy.spatial.cKDTree — SciPy v1.6.1 Reference Guide*. 2021. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.cKDTree.html (visited on 03/07/2021).

[63] *scipy.interpolate.interp1d — SciPy v1.6.1 Reference Guide*. 2021. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html (visited on 03/07/2021).

[64] Trackpy Contributors. *Trackpy: Fast, Flexible Particle-Tracking Toolkit —trackpy 0.4.2 documentation*. 2019. URL: http://soft-matter.github.io/trackpy/v0.4.2/generated/trackpy.motion.compute_drift.html (visited on 03/06/2021).

[65] *Reference Annotations – Cell Tracking Challenge*. Cell Tracking Challenge. 2020. URL: http://celltrackingchallenge.net/annotations/ (visited on 03/08/2021).

[66] Wada, K. *labelme: Image Polygonal Annotation with Python*. 2016.

[67] Amazon Web Services. *Amazon Mechanical Turk*. URL: https://www.mturk.com.

[68] Barrow, H. et al. "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching". *IJCAI*. 1977.

[69] Walt, S. van der et al. *Module: exposure*. 2014.

[70] Trajković, G. *Measurement: Accuracy and Precision, Reliability and ValidityMeasurement: accuracy and precision, reliability and validity. Encyclopedia of Public Health*. Ed. by Kirch, W. Dordrecht: Springer Netherlands, 2008, pp. 888–892.

[71] scikit-learn developers. *Precision-Recall — scikit-learn 0.24.1 documentation*. 2020. URL: `https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html` (visited on 03/11/2021).

[72] The SciPy Community. *numpy.gradient —NumPy v1.20 Manual*. 2020. URL: `https://numpy.org/doc/stable/reference/generated/numpy.gradient.html` (visited on 03/15/2021).

[73] Burden, R. et al. *Numerical Analysis*. Cengage Learning, 2015.

[74] Einstein, A. "Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen". *Annalen der Physik* **322**.8 (1905), pp. 549–560.

[75] Encyclopaedia Britannica, T. E. of. *Brownian motion. Encyclopedia Britannica*. 31, 2017.

[76] Weisstein, E. W. *Wiener Process*. 2021. URL: `https://mathworld.wolfram.com/WienerProcess.html` (visited on 02/19/2021).

[77] Øksendal, B. *Stochastic Differential Equations: An Introduction with Applications*. Hochschultext / Universitext. Springer, 2003.

[78] Google Scholar. *Crocker: Methods of digital video microscopy for... - Google Scholar*. URL: `https://scholar.google.com/scholar?cites=5300714234004324933` (visited on 03/11/2021).

[79] Xing, F. et al. "An Automatic Learning-Based Framework for Robust Nucleus Segmentation". *IEEE Transactions on Medical Imaging* **35**.2 (2016), pp. 550–566.

[80] Chen, Y. et al. "CellTrack R-CNN: A Novel End-To-End Deep Neural Network for Cell Segmentation and Tracking in Microscopy Images" (2021). arXiv: `2102.10377 [cs.CV]`.

[81] Chazal, F. & Michel, B. "An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists" (2017). arXiv: `1710.04019 [math.ST]`.

[82] Otter, N. et al. "A roadmap for the computation of persistent homology". *EPJ Data Science* **6**.1 (2017), p. 17.

[83] Hatcher, A. *Algebraic Topology*. Cambridge University Press, 2001, pp. 97–102.

[84] Topaz, C. M. et al. *Topological Data Analysis of Biological Aggregation Models, Fig 1*. 2015. URL: https://doi.org/10.1371/journal.pone.0126383.g001.

[85] Weisstein, E. W. *Simplex*. 2020. URL: https://mathworld.wolfram.com/Simplex.html.

[86] Weisstein, E. W. *Simplicial Complex*. 2020. URL: https://mathworld.wolfram.com/SimplicialComplex.html.

[87] Hatcher, A. *Algebraic Topology*. Cambridge University Press, 2001, pp. 105–109.

[88] Hatcher, A. *Algebraic Topology*. Cambridge University Press, 2001, pp. 104–107.

[89] Barile, M. & Weisstein, E. W. *Betti Number*. 2020. URL: https://mathworld.wolfram.com/BettiNumber.html.

[90] Hatcher, A. *Algebraic Topology*. Cambridge University Press, 2001.

[91] Ghrist, R. *Elementary Applied Topology*. 1.0. Createspace, 2014.

[92] Ghrist, R. "Barcodes: The persistent topology of data". *Bulletin of the American Mathematical Society* **45**.1 (2008).

[93] Ghrist, R. *Homological Algebra and Data*. 2010. URL: https://www2.math.upenn.edu/~ghrist/preprints/HAD.pdf.

[94] Topaz, C. M. et al. *Topological Data Analysis of Biological Aggregation Models, Fig 4*. 2015. URL: https://doi.org/10.1371/journal.pone.0126383.g004.

[95] Giusti, C. et al. "Clique topology reveals intrinsic geometric structure in neural correlations". *Proceedings of the National Academy of Sciences* **112**.44 (2015), pp. 13455–13460. eprint: https://www.pnas.org/content/112/44/13455.full.pdf.

[96] Topaz, C. M. et al. *Topological Data Analysis of Biological Aggregation Models, Fig 12*. 2015. URL: https://doi.org/10.1371/journal.pone.0126383.g012.

[97] Topaz, C. M. et al. "Topological Data Analysis of Biological Aggregation Models". *PLOS ONE* **10**.5 (2015).

[98] Ulmer, M. et al. "A topological approach to selecting models of biological experiments". *PLOS ONE* **14**.3 (2019), pp. 1–18.

[99] Bauer, U. "Ripser: efficient computation of Vietoris-Rips persistence barcodes" (2019). arXiv: 1908.02518 [math.AT].

[100] Weisstein, E. W. *Cohomology*. 2020. URL: https://mathworld.wolfram.com/Cohomology.html.

[101] Hatcher, A. *Algebraic Topology*. Cambridge University Press, 2001, pp. 190–197.

[102] Silva, V. de et al. "Dualities in persistent (co)homology". *Inverse Problems* **27**.12 (2011), p. 124003.

[103] Saul, N. & Tralie, C. *Scikit-TDA: Topological Data Analysis for Python*. 2019.

[104] Fisher, R. A. "The wave of advance of advantageous genes". *Annals of eugenics* **7**.4 (1937), pp. 355–369.

[105] Thiele, J. C. et al. "Facilitating Parameter Estimation and Sensitivity Analysis of Agent-Based Models: A Cookbook Using NetLogo and R". *Journal of Artificial Societies and Social Simulation* **17**.3 (2014), p. 11.

[106] Dancik, G. M. et al. "Parameter estimation and sensitivity analysis in an agent-based model of Leishmania major infection". *Journal of Theoretical Biology* **262**.3 (2010), pp. 398–412.

[107] Lamperti, F. et al. "Agent-based model calibration using machine learning surrogates". *Journal of Economic Dynamics and Control* **90** (2018), pp. 366–389.

[108] Macklin, P. et al. "Patient-calibrated agent-based modelling of ductal carcinoma in situ (DCIS): From microscopic measurements to macroscopic predictions of clinical progression". *Journal of Theoretical Biology* **301** (2012), pp. 122–140.

[109] Böttger, K. et al. "Investigation of the Migration/Proliferation Dichotomy and its Impact on Avascular Glioma Invasion". *Math. Model. Nat. Phenom.* **7**.1 (2012), pp. 105–135.

[110] Tarantola, A. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005.

[111] Sambridge, M. *An introduction to Inverse Problems*. 2007. URL: `http://web.gps.caltech.edu/classes/ge193.old/lectures/Lecture1.pdf` (visited on 03/03/2021).

[112] Argoul, P. *Overview of Inverse Problems*. DEA. Lecture. 2012. eprint: `https://cel.archives-ouvertes.fr/cel-00781172/file/inverseproblemenglishversion2.pdf`.

[113] Tarantola, A. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005, pp. 1–40.

[114] D'Orsogna, M. R. et al. "Self-Propelled Particles with Soft-Core Interactions: Patterns, Stability, and Collapse". *Phys. Rev. Lett.* **96** (10 2006).

[115] Morse, P. M. "Diatomic Molecules According to the Wave Mechanics. II. Vibrational Levels". *Phys. Rev.* **34** (1 1929), pp. 57–64.

[116] Chuang, Y.-l. et al. "State transitions and the continuum limit for a 2D interacting, self-propelled particle system". *Physica D: Nonlinear Phenomena* **232**.1 (2007), pp. 33–47.

[117] Bhaskar, D. et al. "Analyzing collective motion with machine learning and topology". *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29**.12 (2019), p. 123125.

[118] Bonilla, L. L. et al. "Tracking collective cell motion by topological data analysis". *PLOS Computational Biology* **16**.12 (2020), pp. 1–43.

[119] Kim, W. & Memoli, F. "Spatio-temporal Persistent Homology for Dynamic Metric Spaces" (2019). arXiv: `1812.00949 [math.AT]`.

[120] Kim, W. & Mémoli, F. "Formigrams: Clustering Summaries of Dynamic Data". The University of Manitoba, 2018, pp. 180–188.

[121] Oudot, S. & Solomon, E. "Inverse Problems in Topological Persistence". *Topological Data Analysis*. Ed. by Baas, N. A. et al. Springer International Publishing, 2020, pp. 405–433.

[122] Deissenberg, C. et al. "EURACE: A massively parallel agent-based model of the European economy". *Applied Mathematics and Computation* **204**.2 (2008), pp. 541–552.

[123] Bianchi, F. & Squazzoni, F. "Agent-based models in sociology". *Wiley Interdisciplinary Reviews: Computational Statistics* **7**.4 (2015), pp. 284–306.

[124] Clark, A. *S-I-R Model of Epidemics Part 2*. Dept. of Mechanical Engineering, University of Rochester. 13, 2009. URL: `http://www2.me.rochester.edu/courses/ME406/webexamp6/sir2.pdf` (visited on 02/25/2021).

[125] Weisstein, E. W. *Runge-Kutta Method*. 2021. URL: `https://mathworld.wolfram.com/Runge-KuttaMethod.html` (visited on 02/25/2021).

[126] Platen, E. & Bruti-Liberati, N. *Numerical solution of stochastic differential equations with jumps in finance*. Vol. 64. Springer Science & Business Media, 2010, pp. 246–247.

[127] Dormand, J. & Prince, P. "A family of embedded Runge-Kutta formulae". *Journal of Computational and Applied Mathematics* **6**.1 (1980), pp. 19–26.

[128] *scipy.integrate.ode — SciPy v1.6.1 Reference Guide*. 2021. URL: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html` (visited on 02/25/2021).

[129] Saltelli, A. *Sensitivity analysis in practice : a guide to assessing scientific models*. John Wiley & Sons, 2, 2004.

[130] *scipy.stats.pearsonr — SciPy v1.6.1 Reference Guide*. 2021. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html (visited on 02/26/2021).

[131] Hamby, D. M. "A review of techniques for parameter sensitivity analysis of environmental models". *Environmental monitoring and assessment* **32**.2 (1994), pp. 135–154.

[132] Saul, N. & Tralie, C. *Ripser demonstration — Ripser.py 0.6.0 documentation*. 2019. URL: https://ripser.scikit-tda.org/en/latest/notebooks/Basic%5C%20Usage.html#Specify-which-homology-classes-to-compute (visited on 03/01/2021).

[133] Hatcher, A. *Algebraic Topology*. Cambridge University Press, 2001, pp. 130–131.

[134] Jin, W. et al. "Stochastic simulation tools and continuum models for describing two-dimensional collective cell spreading with universal growth functions". *Physical Biology* **13**.5 (2016).

[135] Lagergren, J. H. et al. "Biologically-informed neural networks guide mechanistic modeling from sparse experimental data". *PLOS Computational Biology* **16** (2020), pp. 1–29.

[136] Hatcher, A. *Algebraic Topology*. Cambridge University Press, 2001, pp. 110–113.

[137] Tarantola, A. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, 2005, pp. 62–80.

[138] Weisstein, E. W. *Least Squares Fitting*. 2021. URL: https://mathworld.wolfram.com/LeastSquaresFitting.html (visited on 03/03/2021).

[139] Cameron, A. & Trivedi, P. *Regression Analysis of Count Data*. Econometric Society Monographs. Cambridge University Press, 2013.

[140] Weisstein, E. W. *Nonlinear Least Squares Fitting*. 2021. URL: https://mathworld.wolfram.com/NonlinearLeastSquaresFitting.html (visited on 03/03/2021).

[141] Davison, A. *Statistical Models*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2003.

[142] Lewis, R. M. et al. "Direct search methods: then and now". *Journal of Computational and Applied Mathematics* **124**.1 (2000). Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations, pp. 191–207.

[143] Nelder, J. A. & Mead, R. "A simplex method for function minimization". *The computer journal* **7**.4 (1965), pp. 308–313.

[144] *Find minimum of unconstrained multivariable function using derivative-free method - MATLAB fminsearch*. 2021. URL: https://www.mathworks.com/help/matlab/ref/fminsearch.html#bvadxhn-12 (visited on 03/04/2021).

[145] *minimize(method='Nelder-Mead') — SciPy v1.6.1 Reference Guide*. 2021. URL: https://docs.scipy.org/doc/scipy/reference/optimize.minimize-neldermead.html (visited on 03/05/2021).

[146] Casella, G. & Berger, R. *Statistical Inference*. Cengage Learning, 2021, pp. 529–580.

[147] Box, M. "A new method of constrained optimization and a comparison with other methods". *The Computer Journal* **8**.1 (1965), pp. 42–52.

[148] Mara, P. S. "Triangulations for the cube". *Journal of Combinatorial Theory, Series A* **20**.2 (1976), pp. 170–177.

[149] *Simplexity of the n-cube*. 6, 2008. URL: http://garden.irmacs.sfu.ca/?q=op/simplexity_of_the_cube (visited on 02/21/2021).

[150] Weisstein, E. W. *Hypercube Graph*. 2021. URL: https://mathworld.wolfram.com/HypercubeGraph.html.

[151] Box, G. E., Draper, N. R., et al. *Empirical model-building and response surfaces*. Vol. 424. Wiley New York, 1987.

[152] Zhang, S. et al. "GPU-Accelerated Computation of Vietoris-Rips Persistence Barcodes" (2020). arXiv: 2003.07989 [cs.CG].

[153] Cohen-Steiner, D. et al. "Vines and Vineyards by Updating Persistence in Linear Time". *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*. SCG '06. Sedona, Arizona, USA: Association for Computing Machinery, 2006, pp. 119–126.

[154] Gelman, A. et al. *Bayesian data analysis*. CRC press, 2013.

[155] Villee, C. A. *Morphology. Encyclopedia Britannica*. 21, 2018.

[156] Thermo Fisher Scientific. *Cell Morphology —Thermo Fisher Scientific - US*. 2021. URL: https://www.thermofisher.com/us/en/home/references/gibco-cell-culture-basics/cell-morphology.html (visited on 03/09/2021).

[157] Abercrombie, M. & Heaysman, J. E. "Observations on the social behaviour of cells in tissue culture: I. Speed of movement of chick heart fibroblasts in relation to their mutual contacts". *Experimental Cell Research* **5**.1 (1953), pp. 111–131.

[158] Ranft, J. et al. "Fluidization of tissues by cell division and apoptosis". *Proceedings of the National Academy of Sciences* **107**.49 (2010), pp. 20863–20868. eprint: https://www.pnas.org/content/107/49/20863.full.pdf.

[159] Petrie, R. J. et al. "Random versus directionally persistent cell migration". *Nature Reviews Molecular Cell Biology* **10**.8 (2009), pp. 538–549.

[160] Welf, E. S. et al. "Migrating fibroblasts reorient directionality by a metastable, PI3K-dependent mechanism". *Journal of Cell Biology* **197**.1 (2, 2012), pp. 105–114.

[161] Ulmer, M. et al. *A topological approach to selecting models of biological experiments, Figure 2*. 2019. URL: https://doi.org/10.1371/journal.pone.0213679.g002.

[162] Haugh, J. & Baldwin, S. Personal Communication. Research discussion on the viability of the D'Orsogna model for scratch assay data. 18, 2020.

[163] *tf.keras.utils.plot_model —TensorFlow Core v2.4.1*. 2021. URL: https://www.tensorflow.org/api_docs/python/tf/keras/utils/plot_model (visited on 02/27/2021).

[164] Hastie, T. et al. *The Elements of Statistical Learning*. 2nd ed. New York, NY: Springer-Verlag, 2009.

APPENDICES

Appendix A

Centroid Detector Architecture

While the body text includes the relevant particular of the architecture and training of our centroid detector, we include a diagram of the network here for the sake of transparency and reproducibility. This diagram is generated automatically using the model architecture plotting utility included in TensorFlow's Keras module [163].

Note that the long lines indicate the concatenation of downsampled feature maps to upsampled maps of the same size, which is the hallmark of U-Net-type architectures. The layer type and output dimensions are shown in each box, illustrating the standard approach of contracting map size and increasing depth at higher levels of abstraction. Finally, note that the output map has the same dimensions as the input image, providing a one-to-one correspondence between image pixel and p-map pixel.

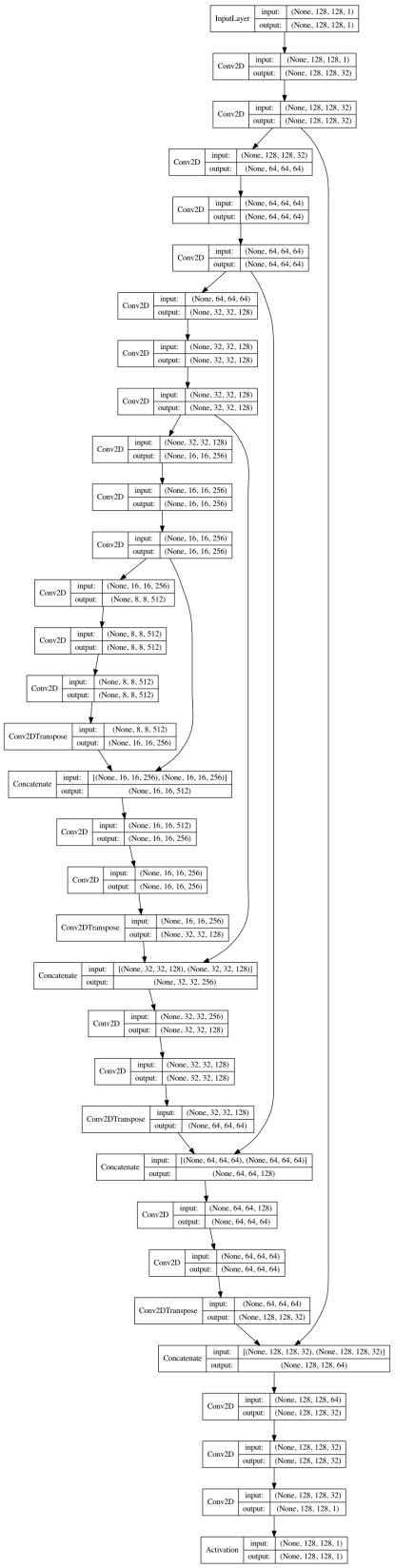

Figure A.1: Diagram of the centroid detector neural network architecture.

Appendix B

Additional Plots: Chapter 4

B.1 Trench Plots for Time Span Analysis

We include additional trench plots in this appendix, showing that they illustrate roughly the same behavior over large time spans for each of the three model parameters.

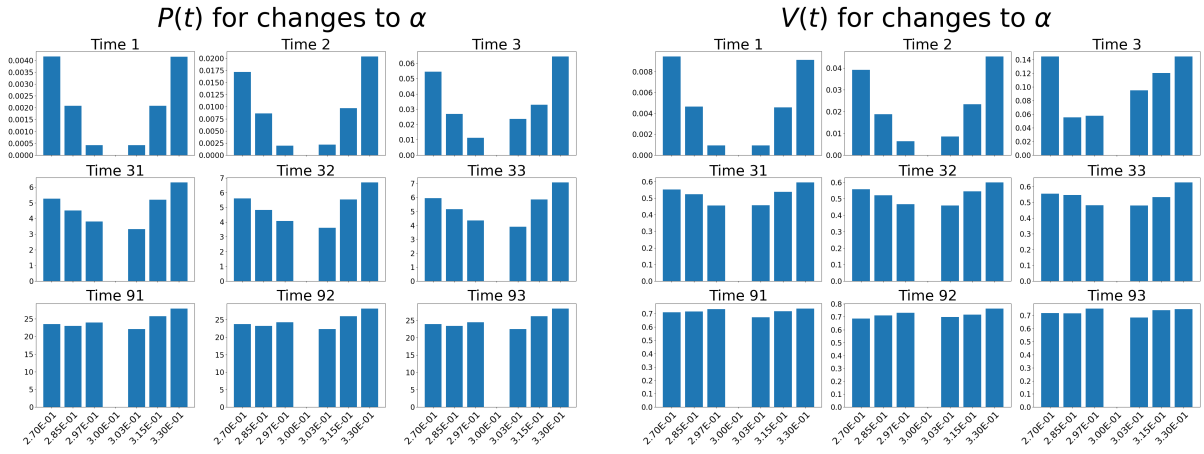

Figure B.1: A selection of trench plots for the positions (left) and velocities (right) of the deterministic model for changes to α.

B.2 Pearson r Plots for Time Span Analysis

We show the plots of the Pearson correlation coefficients r for the third model parameter ℓ below.

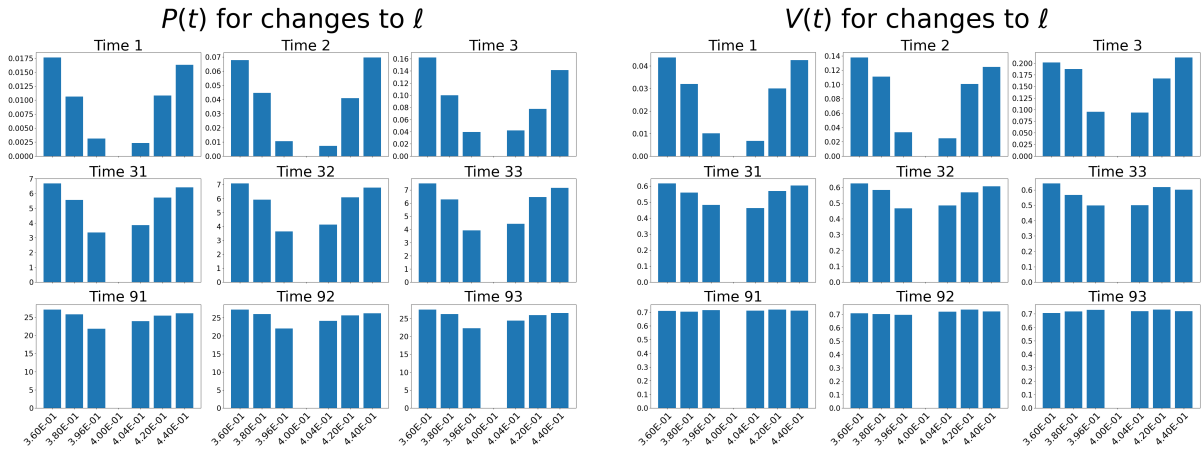

Figure B.2: A selection of trench plots for the positions (left) and velocities (right) of the deterministic model for changes to ℓ.

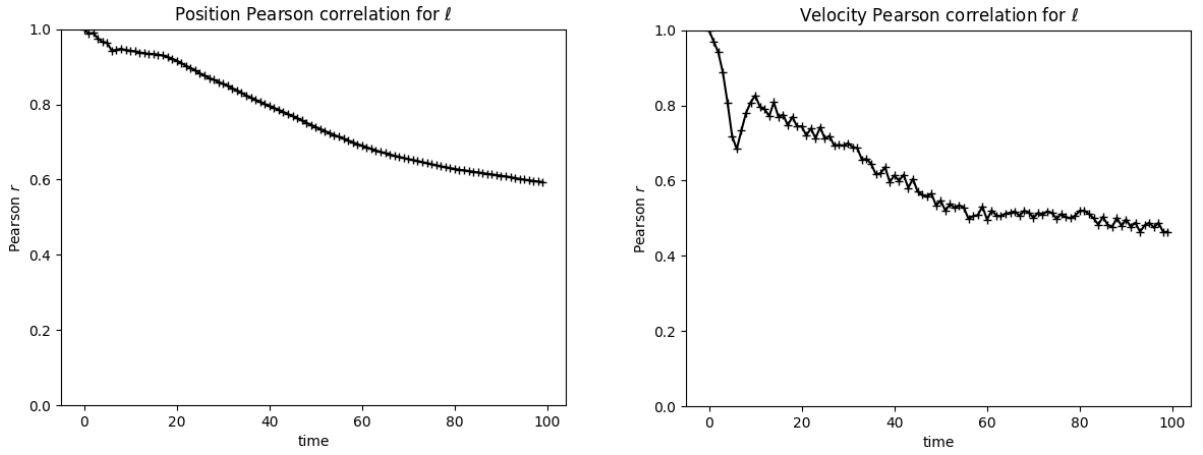

Figure B.3: Plots of the Pearson correlation coefficients for each time step of the position (left) and velocity (right) trench plots for the model parameter ℓ.

B.3 Plots for Stochasticity Analysis

This section includes the trench plots with the minimum level of noise ($\sigma = 10^{-3}$) for changes to each model parameter, the velocity correlation plot for C with a moderate level of noise ($\sigma = 10^{-2}$), and the trench plots for C with an overwhelming level of noise ($\sigma = 10^{-1}$).

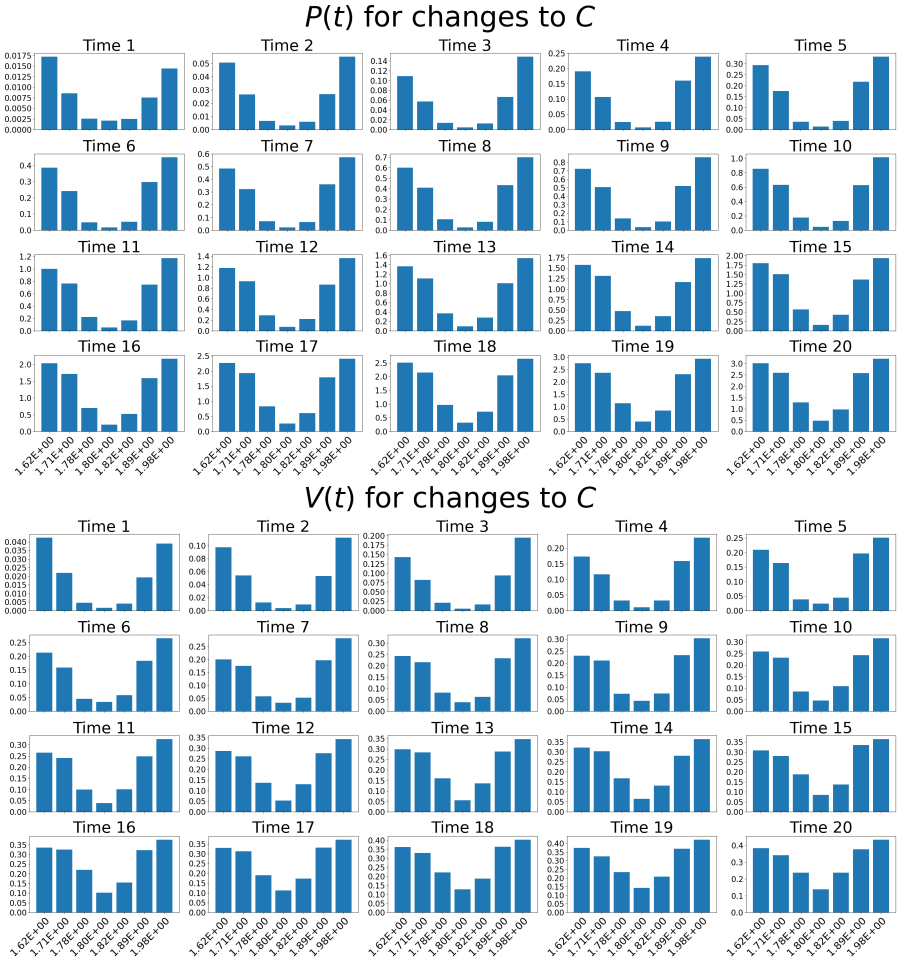

Figure B.4: Trench plots for $\sigma = 10^{-3}$ illustrating the clear correlation for the C model parameter (Figure 4.7, column 2).

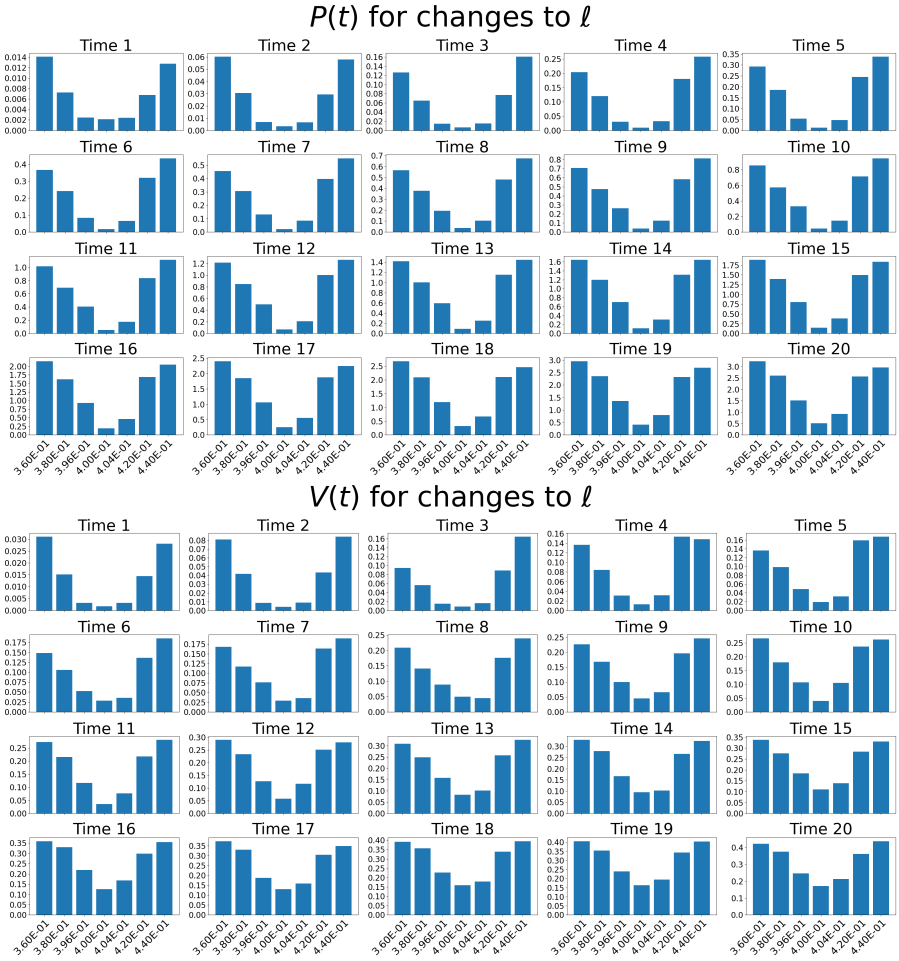

Figure B.5: Trench plots for $\sigma = 10^{-3}$ illustrating the clear correlation for the ℓ model parameter (Figure 4.7, column 3).

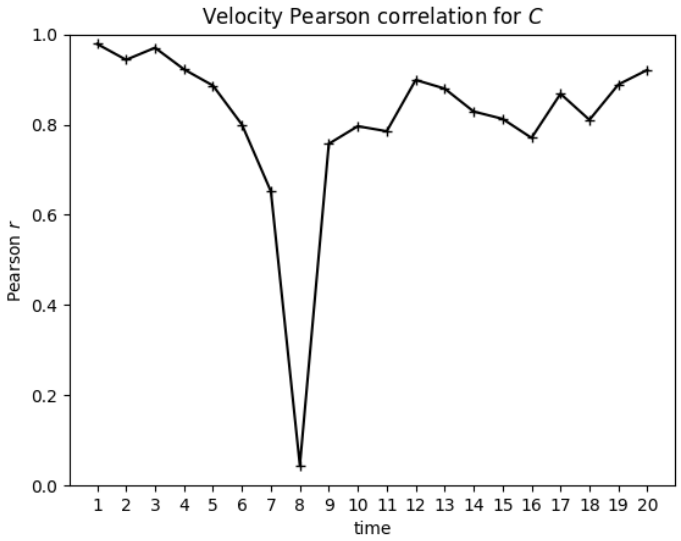

Figure B.6: Correlation plots for C from data where $\sigma = 10^{-2}$.

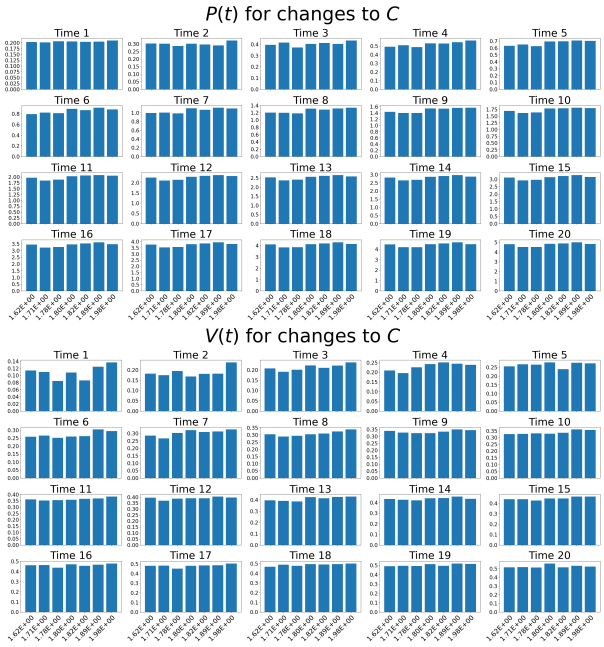

Figure B.7: Trench plots for C from data where $\sigma = 10^{-1}$ demonstrating a complete lack of correlation.