Mathematical modeling and simulation of protrusion-driven cell dynamics

---

Daniel Schindler

DISSERTATION

Zur Erlangung des akademischen Grades
“doctor rerum naturalium” (Dr. rer. nat.)
in der Wissenschaftsdisziplin “Mathematik”

eingereicht an der
MATHEMATISCH-NATURWISSENSCHAFTLICHEN FAKULTÄT
INSTITUT FÜR MATHEMATIK
DER UNIVERSITÄT POTSDAM

Potsdam, der 11. November 2023

Dedicated to my dear friend Yaron Tausky,

who passed away during the writing of this thesis.

Daniel Schindler
Mathematical modeling and simulation of protrusion-driven cell dynamics
Dissertation, Potsdam, der 11. November 2023

Supervisors:	Prof. Dr. Matthias Holschneider
	Prof. Dr. Wilhelm Huisinga
Mentor:	Prof. Dr. Carsten Beta
Reviewers:	Prof. Dr. Matthias Holschneider (University of Potsdam)
	Prof. Dr. Till Bretschneider (University of Warwick)
	Prof. Marc Hoffmann, PhD (Paris-Dauphine University)
Universität Potsdam:	Angewandte Mathematik
	Mathematische Modellierung und Systembiologie
	Institut für Mathematik
	Mathematisch-Naturwissenschaftliche Fakultät
	Karl-Liebknecht-Str. 24/25
	14476 Potsdam

Abstract

Amoeboid cell motility takes place in a variety of biomedical processes such as cancer metastasis, embryonic morphogenesis, and wound healing. In contrast to other forms of cell motility, it is mainly driven by substantial cell shape changes. Based on the interplay of explorative membrane protrusions at the front and a slower-acting membrane retraction at the rear, the cell moves in a crawling kind of way. Underlying these protrusions and retractions are multiple physiological processes resulting in changes of the cytoskeleton, a meshwork of different multi-functional proteins. The complexity and versatility of amoeboid cell motility raise the need for novel computational models based on a profound theoretical framework to analyze and simulate the dynamics of the cell shape.

The objective of this thesis is the development of (i) a mathematical framework to describe contour dynamics in time and space, (ii) a computational model to infer expansion and retraction characteristics of individual cell tracks and to produce realistic contour dynamics, (iii) and a complementing Open Science approach to make the above methods fully accessible and easy to use.

In this work, we mainly used single-cell recordings of the model organism *Dictyostelium discoideum*. Based on stacks of segmented microscopy images, we apply a Bayesian approach to obtain smooth representations of the cell membrane, so-called cell contours. We introduce a one-parameter family of regularized contour flows to track reference points on the contour (virtual markers) in time and space. This way, we define a coordinate system to visualize local geometric and dynamic quantities of individual contour dynamics in so-called kymograph plots. In particular, we introduce the local marker dispersion as a measure to identify membrane protrusions and retractions in a fully automated way.

This mathematical framework is the basis of a novel contour dynamics model, which consists of three biophysiologically motivated components: one stochastic term, accounting for membrane protrusions, and two deterministic terms to control the shape and area of the contour, which account for membrane retractions. Our model provides a fully automated approach to infer protrusion and retraction characteristics from experimental cell tracks while being also capable of simulating realistic and qualitatively different contour dynamics. Furthermore, the model is used to classify two different locomotion types: the amoeboid and a so-called fan-shaped type.

With the complementing Open Science approach, we ensure a high standard regarding the usability of our methods and the reproducibility of our research. In this context, we introduce our software publication named `AmoePy`, an open-source Python package to segment, analyze, and simulate amoeboid cell motility. Furthermore, we describe measures to improve its usability and extensibility, e.g., by detailed run instructions and an automatically generated source code documentation, and to ensure its functionality and stability, e.g., by automatic software tests, data validation, and a hierarchical package structure.

The mathematical approaches of this work provide substantial improvements regarding the modeling and analysis of amoeboid cell motility. We deem the above methods, due to their generalized nature, to be of greater value for other scientific applications, e.g., varying organisms and experimental setups or the transition from unicellular to multicellular movement. Furthermore, we enable other researchers from different fields, i.e., mathematics, biophysics, and medicine, to apply our mathematical methods. By following Open Science standards, this work is of greater value for the cell migration community and a potential role model for other Open Science contributions.

Zusammenfassung

Amöboide Zellmotilität findet bei einer Vielzahl biomedizinischer Prozesse wie Krebsmetastasierung, embryonaler Morphogenese und Wundheilung statt. Im Gegensatz zu anderen Formen der Zellmotilität wird sie hauptsächlich durch erhebliche Formveränderungen der Zelle angetrieben. Sie beruht auf dem Zusammenspiel von explorativen Membranausstülpungen an der Vorderseite und einem langsamer wirkenden Membraneinzug an der Rückseite. Die Komplexität amöboider Zellmotilität machen neue Berechnungsmodelle erforderlich, um die Dynamik der Zellform mathematisch fundiert zu analysieren und zu simulieren.

Ziel dieser Arbeit ist die Entwicklung (i) eines mathematischen Frameworks zur Beschreibung der Konturendynamik in Zeit und Raum, (ii) eines Computermodells, um Eigenschaften der Membranveränderungen von einzelnen Zellen zu inferieren und gleichzeitig realistische Konturdynamiken zu simulieren, (iii) und eines ergänzenden Open-Science-Ansatzes, um die oben genannten Methoden vollständig zugänglich und leicht anwendbar zu machen.

Auf der Grundlage von aufeinander folgenden Mikroskopiebildern vom Modellorganismus *Dictyostelium discoideum*, wenden wir einen Bayesschen Ansatz an, um glatte Darstellungen der Zellmembran, sogenannte Zellkonturen, zu erhalten. Wir führen eine einparametrige Familie von regularisierten Konturflüssen ein, um Referenzpunkte auf der Kontur (virtuelle Marker) in Zeit und Raum zu verfolgen. Auf diese Weise definieren wir ein Koordinatensystem zur Visualisierung lokaler geometrischer und dynamischer Größen der individuellen Konturdynamiken in sogenannten Kymographen-Plots. Insbesondere führen wir die lokale Marker-Dispersion ein, mit der signifikante Membranveränderungen identifiziert werden können.

Dieses mathematische Framework bildet die Grundlage für unser neues Modell zur Beschreibung von Konturendynamiken. Es besteht aus drei biophysiologisch motivierten Komponenten: einem stochastischen Term, der die Membranausstülpungen steuert, und zwei deterministischen Termen, die das Membraneinziehen, unter Berücksichtigung der Konturform und -fläche, steuern. Unser Modell bietet einen vollautomatisierten Ansatz zur Inferrenz der Charakteristiken von Membranveränderungen für experimentelle Zelldaten. Außerdem ermöglicht es die Simulation von realistischen und qualitativ unterschiedlichen Konturendynamiken.

Mit dem ergänzenden Open-Science-Ansatz setzen wir einen hohen Standard hinsichtlich der Nutzbarkeit unserer Methoden und der Reproduzierbarkeit unserer Forschung. In diesem Kontext stellen wir die Softwarepublikation `AmoePy` vor, ein Open-Source-Pythonpaket zur Segmentierung, Analyse und Simulation von amöboider Zellmotilität. Darüber hinaus beschreiben wir Maßnahmen zur Verbesserung der Benutzerfreundlichkeit und Erweiterbarkeit, z. B. durch detaillierte Ausführanweisungen und eine automatisch generierte Quellcodedokumentation, und zur Gewährleistung der Funktionalität und Stabilität, z. B. durch automatische Softwaretests, Datenvalidierung und eine hierarchische Paketstruktur.

Die mathematischen Methoden dieser Arbeit stellen wesentliche Verbesserungen in der Modellierung und Analyse der amöboiden Zellmotilität dar. Wir sind der Ansicht, dass die oben genannten Methoden aufgrund ihrer Verallgemeinerbarkeit von größerem Wert für andere wissenschaftliche Anwendungen sind und potentiell einsetzbar in verschiedenen Wissenschaftsfeldern sind, u. a. Mathematik, Biophysik und Medizin. Durch die Einhaltung von Open-Science-Standards ist diese Arbeit von größerem Wert und ein potenzielles Vorbild für andere Open-Science-Beiträge.

Acknowledgments

During my doctoral studies and the process of writing this thesis, the following people have provided me with decisive scientific and personal support.
First, I would like to extend my sincere thanks to my advisor Matthias Holschneider for the many fruitful discussions, his exceptional knowledge, the insightful moments I had while working with him, and for giving me the freedom to pursue my own scientific interests. Furthermore, I would like to express my deepest gratitude to my advisor Wilhelm Huisinga for his scientific guidance and strong support, the regular meetings, the productive and joyful work environment, and his endless trust in my work. I am also very grateful to Carsten Beta for his extensive experience, especially in the field of biophysics, his helpful contributions and suggestions, and his great devotion to the project B02.
I would like to thank Ted Moldenhawer for his constructive scientific feedback and helpful contributions, the many hours of collecting the data and testing our software package `AmoePy`, and his attentiveness and friendship throughout our PhD studies. Many thanks go to Hendrik Gessner for his help in conceptualizing `AmoePy` and his advising role during the development process. Furthermore, I am very thankful to Lena Lindenmeier for her great support regarding the graphical user interface and the automatic testing of `AmoePy`. Gratefully, I want to acknowledge Victor Lesur for his work on the image segmentation routines.
Also, I would like to thank the entire CRC1294 for supporting my research and providing me with valuable scientific input through many colloquia and seminars. The collaboration with other scientists from different disciplines was very helpful and I am thankful for the many joyful moments, especially, at the CRC spring schools and jamborees.
I am very grateful to the present and former members of the systems biology working group: Niklas Hartung, Corinna Maier, Sabine Stübler, Undine Falkenhagen, Daniel Seeler, Franziska Thoma, Elena Pascual Garcia, and Andreas Braunß. They provided me with insightful suggestions and it was a great pleasure to work with them. A special thanks goes to Bertram Arnold for his mathematical support throughout the years. Furthermore, I want to thank Christian Riedel for his valuable Feedback on my Open Science chapter.
Many thanks go to Ruben Dewicke, Anne Lange, and Catrin Klein for joyful hours of co-working and the amazing personal support. A special thanks goes to the HTW Tonkollektiv and the Kammerchor Bethanien for giving me valuable feedback regarding the installation process of `AmoePy`, but more importantly, for the strong vocal and mental support over the years. Finally, I am deeply grateful to Katharina Grosch, my parents Karl-Michael Schindler and Martina Hund-Schindler, and my brothers Alexander and Benjamin Schindler for encouraging me, believing in me, and giving me the strength and confidence to write this thesis.

Publications

This thesis contains parts that were published or are currently in the process of being published in the following original research articles:

Sections 3.1 and 4:

[DS1]: Daniel Schindler, Ted Moldenhawer, Maike Stange, Valentino Lepro, Carsten Beta, Matthias Holschneider, and Wilhelm Huisinga. "Analysis of protrusion dynamics in amoeboid cell motility by means of regularized contour flows". In: *PLoS Computational Biology* 17.8 (2021), e1009268. DOI: 10.1371/journal.pcbi.1009268

Sections 2.2, 3.2, and 5:

[DS2]: Daniel Schindler, Ted Moldenhawer, Carsten Beta, Wilhelm Huisinga, and Matthias Holschneider. "Three-component contour dynamics model to simulate and analyze amoeboid cell motility". In: *arXiv* (2022). DOI: 10.48550/arXiv.2210.12978

All results and figures of the above articles are fully reproducible via the following software publication:

Section 6:

[DS3]: Daniel Schindler, Ted Moldenhawer, Lena Lindenmeier, Victor Lesur, and Matthias Holschneider. *AmoePy: A Python-based toolbox for analyzing and simulating amoeboid cell motility*. Zenodo, Software. 2020. DOI: 10.5281/zenodo.3982371

Contribution to the above articles: Collectively, all co-authors conceptualized the principle research ideas and discussed the resulting findings. The author of this dissertation realized the specific research objectives, conducted the research, analyzed and interpreted the findings, and wrote the first manuscript drafts. The co-author critically revised the manuscript and proposed various amendments.

Contribution to the software publication: The author of this dissertation conceptualized and developed most of the source code. For each source code file, the individual authorship is listed in the file's header.

Introduction and Outlook:

[DS4]: Ted Moldenhawer, Eduardo Moreno, Daniel Schindler, Sven Flemming, Matthias Holschneider, Wilhelm Huisinga, Sergio Alonso, and Carsten Beta. "Spontaneous transitions between amoeboid and keratocyte-like modes of migration". In: *Front. Cell Dev.* 10:898351 (2022). DOI: 10.3389/fcell.2022.898351

Contribution to the last article: The author of this thesis implemented the majority of the underlying computational methods, contributed ideas to the specific research objectives, made suggestions regarding the text and figures, and revised the manuscript.

Contents

List of Figures

List of Tables

Acronyms

Abbreviation	Description	Page
AAF	area adjustment flow	27
APCSF	area-preserving curve-shortening flow	5
BSD	Berkeley Software Distribution	92
BY	by attribution	92
cAMP	cyclic adenosine monophosphate	7
CC	Creative Commons	91
CI/CD	continuous integration/continuous delivery	99
CIL	Cell Image Library	104
CLI	command line interface	97
CMSO	Cell Migration Standardisation Organisation	103
CRC	collaborative research center	89
CSF	curve-shortening flow	5
DFG	Deutsche Forschungsgemeinschaft	99
DOI	digital object identifier	93
F-actin	filamentous actin	10
G-actin	globular actin	10
GNU GPL	GNU General Public License	92
GP	Gaussian process	11
GPR	Gaussian process regression	3
GUI	graphical user interface	94
HP	Hawkes process	5
ISA	Investigation/Study/Assay	103
MIACME	Minimum Information About a Cell Migration Experiment	103
MLE	maximum likelihood estimation	17
NC	non-commercial	92
ND	no derivative	92

Abbreviation	Description	Page
OME	Open Microscopy Environment	93
ORCID	Open Researcher and Contributor ID	103
OSF	Open Science Framework	103
OTN	Open Traits Network	103
OUP	Ornstein-Uhlenbeck process	5
RCFM	regularized contour flow method	107
SA	share-alike	92
SDE	stochastic differential equation	29

Introduction 1

Being one of the most widespread forms of cell motility, amoeboid motion takes place in a variety of biological processes. As part of the immune system response of the human body, amoeboid migrating neutrophils play a key role in eliminating pathogenic bacteria or fungi [1]. Other amoeboid migrating cells, being essential to the humane immune response, include lymphocytes, natural killer cells, T cells and monocytes [2]. For this reason, investigating amoeboid cell motility is crucial to understand medical processes, e.g., the healing of wounded tissues [3], but also the invasion and metastasis of cancer [4]. For example, leukemia, lymphoma, and small cell lung carcinoma migrate in an amoeboid way [5], which underlines the high scientific relevance of this field.

Amoeboid cell motility is characterized by dynamical changes of the the cell shape. Through a coordinated pattern of explorative protrusions at the front and slower acting retractions at the rear, the cell is moving in a crawling kind of way [6, 7]. The mechanical forces which drive amoeboid motion are resulting from changes of the actin cytoskeleton, a meshwork of multifunctional proteins within the cell. By polymerization and depolymerization of actin filaments, parts of the cell membrane either expand or retract. In numerous cases, amoeboid motion and the underlying physiological processes have been studied based on experimental recordings of the model organism *Dictyostelium discoideum*, a slime mold with various advantageous properties [8, 9].

Current modeling approaches differ in complexity and dimensionality, addressing different aspects of amoeboid cell motility, e.g., random walk models to predict the center of mass trajectory of the cell [10, 11, 12], or higher-dimensional models to describe the underlying physiological processes and mechanics such as actin protrusion dynamics [13, 14, 15]. However, many models remain qualitative, where, oftentimes, predicted model outcomes are compared to experimental data by visual inspection only. On the other hand, a model which aims to comprise all physiological processes and mechanistic forces, is difficult to realize from a computational standpoint, but is also difficult to interpret, due to a large number of model parameters and entangled subprocesses [15]. This raises a need of quantitative, data-driven, and fully automated approaches to identify and analyze the key characteristics of amoeboid cell migration, to classify different locomotion types, and to predict realistic motility patterns. Ideally, these approaches are intuitively comprehensible and rely on well-defined and, especially, well-understood mathematical concepts. Often, the underlying data and source codes are not fully documented or not even published, which makes it hard, if not impossible, for other researchers to use these models and to apply them for different scientific applications. Hence, the development of a computational amoeboid cell motility model should be complemented with an

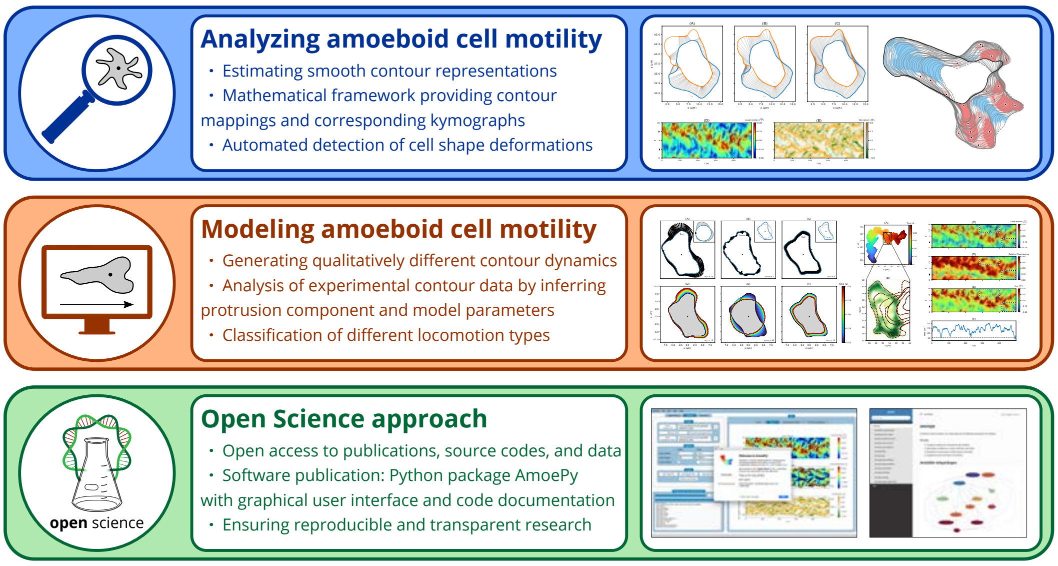

Analyzing amoeboid cell motility
• Estimating smooth contour representations
• Mathematical framework providing contour mappings and corresponding kymographs
• Automated detection of cell shape deformations
Modeling amoeboid cell motility
• Generating qualitatively different contour dynamics
• Analysis of experimental contour data by inferring protrusion component and model parameters
• Classification of different locomotion types
Open Science approach
• Open access to publications, source codes, and data
• Software publication: Python package AmoePy with graphical user interface and code documentation
• Ensuring reproducible and transparent research
open science

Figure 1.1. Summary of main scientific objectives to be addressed in this thesis.

Open Science strategy to ensure the reproducibility of all scientific results and the reusability of all mathematical methods.

In this thesis, we address these issues with a systematic, quantitative, and data-driven approach to analyze and simulate amoeboid cell motility. In Fig 1.1, we give a short overview of this approach, which is separated into three main topics. It is structured in the following way:

(i) a mathematical framework for the description of experimentally observed cell contour dynamics,

(ii) a computational model to infer key motility characteristics of experimental data and to predict realistic contour dynamics,

(iii) and a complementing Open Science approach to ensure the reproducibility of our research and the reusability of our methods.

Analyzing amoeboid cell motility. Various approaches to analyze amoeboid cell motility are based on 3D time-lapse images to incorporate the cell's ability to move likewise in all directions [16, 17]. However, these 3D approaches are more expensive from a computational point of view and experimentally very challenging. For this reason, the majority of methods to analyze cell migration is based on microscopy imaging data of 2D cross-sections of the cell [18, 19, 20]. Stacks of segmented microscopy images are then used to obtain spatio-temporal data of the cell boundary, so-called cell contours. To parametrize the cell contour, the contour arc length is used [18, 21, 19] as well as polar coordinates relative to the center of mass [22]. Most commonly, the cell contour is modeled as a "rigid chain" described by a fixed number of equidistant segmentation points [21, 19]. Alternatively, cell contours are modeled as "elastic chains", where the number of segmentation points and the distances between neighboring points vary over time [18]. As a next step, the dynamics of these cell contours are studied with the aim to identify key characteristics of amoeboid cell migration, i.e., substantial membrane protrusions or retractions [20, 23, 24, 25, 26]. Noteworthy, the mapping of consecutive cell contours in time and space is intrinsically not well defined. Different methods have been established to address this issue, where, in most cases, a fixed number of reference points on the contour, so-called virtual markers, is tracked in time and space [27, 24], e.g., by using electric field equations [28] or mechanistic spring equations [29]. Based on these virtual marker trajectories, different geometric and dynamic quantities can be visualized and studied in form of so-called kymograph plots [7, 21, 24].

In this thesis, we use a Gaussian process regression (GPR), a Bayesian regression framework with a kernel-based covariance structure, to obtain smooth representations of the cell membrane. The corresponding curvature of the contour can be analytically computed as a by-product of the GPR. As an alternative approach to track virtual markers in time and space, we propose a mathematical framework and computational method, which is based on a one-parameter family of regularized flows. Based on these regularized flows, we define a spatio-temporal coordinate system of experimental contour dynamics to obtain kymograph visualizations of different quantities of interest, e.g., the local motion and the contour curvature. In this context, we define a novel dynamic quantity, the local dispersion, a stretching rate of neighboring virtual markers, which we use to identify substantial membrane protrusions and retractions of different intensities in a fully automated away. The above method was mainly applied to the social amoeba *D. discoideum*, but also to early embryonic killifish cells (*Fundulus heteroclitus*), keratocytes cultured from Central American cichlids (*Hypsophrys nicaraguensis*) and artificially generated academic test cases. In a follow-up article, we applied the above method successfully to mutated *D. discoideum* cells, where we detected switches between two distinct motility types: the amoeboid and a so-called fan-shaped type [DS4].

Modeling amoeboid cell motility. So far, a variety of different approaches to model amoeboid cell motility have been developed. These models differ in complexity and incorporate different aspects of amoeboid cell motility, e.g., the dynamics of the cell's centroid [10, 30, 11, 31, 32, 33], reaction-diffusion models to describe the dynamics of different biochemical compounds [34, 35, 36, 37] and the influence of external chemoattractants [38, 39, 40, 41], and mechanistic models focusing on the interplay of different forces affecting the cell boundary [42, 43, 15]. Despite the large number of varying computational models, different challenging problems remain open, e.g., the inference of key motility characteristics or the *a priori* specification of the parameter regime to generate specific motility patterns. Only few approaches provide a full integration of experimental data into the model [44]. Furthermore, the underlying parameter estimation is oftentimes based on a sensitivity analysis and, thus, difficult to perform for a large number of model parameters [15, 45, 46, 47].

In this thesis, we propose a novel three-component contour dynamics model to simulate and analyze amoeboid cell motility. It comprises three biophysiologically motivated components: a stochastic term, accounting for membrane protrusions, and two deterministic terms, accounting for membrane retractions by controlling the shape and area of the contour. The stochastic term is driven by a Hawkes process, a self-exciting Poisson point process, whereas the two geometric terms are based on the area-preserving curve-shortening flow and an area adjustment flow. Our model is capable of generating a variety of qualitatively different and, especially, realistic contour dynamics. Moreover, the model allows to analyze experimental contour dynamics by inferring key motility characteristics, e.g., the protrusion component driving the cell. Due to the low model complexity, the underlying model parameters can be estimated in a fast and straightforward way. Finally, based on the estimates of the different model weights, we classify two different locomotion types: the amoeboid and the fan-shaped type, both experimentally recorded for *D. discoideum*.

Open Science approach and reproducibility of science. In recent years, the issue of research reproducibility became more and more important. Some authors and researchers even speak of a reproducibility or credibility crisis in science [48, 49, 50, 51], whereas others disagree with the narrative of a declining research quality [52]. Nevertheless, the sharing of scientific methods, data, and source codes was never as easy as now, bearing the great potential to increase research reproducibility, to accelerate the knowledge transfer in general, and to improve the scientific quality altogether [53, 54, 55, 56]. For this reason, research reproducibility is intertwined with the concept of Open Science, a loose collection of different ideas and principles with the goal to make scientific work more transparent and to increase its public outreach.

In the same spirit, we developed an Open Science approach complementing the mathematical methods described in this thesis. As part of this Open Science approach, we provide data and software publications to ensure a reproducibility of our findings and to facilitate the usage of our methods for researchers of different Disciplines, e.g., medicine, biophysics, and biostatistics. In this context, we developed `AmoePy`, an open source software package based on Python to segment, analyze, and simulate amoeboid contour dynamics. Furthermore, we took measures to (1) increase its usability, e.g., by multiple documentation files, a simple installation process, and a graphical user interface, and (2) ensure its functionality and stability, e.g., by automatic software tests and a hierarchical package structure. We envision `AmoePy` to be of greater value for the cell migration community and to be a role model for other Open Science contributions.

Outline. First, we provide a short biophysical background in Chapter 2 about the model organism terminology, the reasons why *D. discoideum* is considered to be such a model organism, and amoeboid cell motility in general.

In Chapter 3, we provide the mathematical background to this thesis. We start with general information about the GPR, its usage to obtain smooth cell contours, to compute the corresponding contour curvature, and its application to obtain spatio-temporal 3D contour representations called “amoeba tubes”. Furthermore, we introduce geometric contour flows, e.g., the curve-shortening flow (CSF) and the area-preserving curve-shortening flow (APCSF), and two stochastic processes, the Ornstein-Uhlenbeck process (OUP) and the Hawkes process (HP), which are later used in our three-component contour dynamics model.

In Chapter 4, we introduce a mathematical framework based on a one-parameter family of regularized contour flows to analyze amoeboid cell motility. First, we explain how the experimental contour data was acquired. Then, we describe the underlying mathematics of mapping consecutive cell contours in time and space. By doing so, we give rise to different choices of spatio-temporal coordinate systems, resulting from these mappings, to analyze contour dynamics. In a next step, this framework is applied to experimental contour data, recorded for *D. discoideum*, to identify and characterize main protrusion and retraction events based on a newly introduced quantity called local dispersion. These identified protrusion and retraction events are then analyzed with respect to different statistical quantities, e.g., area, growth time, and direction. In a final step, the approach is applied to contour dynamics for other organisms, i.e., *Fundulus heteroclitus Hypsophrys nicaraguensis*, and different experimental setups, i.e., bright-field microscopy, fluorescence microscopy, and varying frame rates.

Different modeling approaches of amoeboid cell motility are summarized at the beginning of Chapter 5. We, then, propose our three-component contour dynamics to simulate and analyze amoeboid cell motility. In this context, we introduce the general notations, geometric flows, and stochastic processes, on which our model is based on. Furthermore, we describe the inference approach of our model and how the underlying parameters are estimated. A variety of qualitatively different cell tracks are then generated with our model. Finally, the model is used to analyze experimental contour dynamics, recorded for *D. discoideum*, by inferring the protrusion component and estimating the parameters of the model. Based on this parameter estimation, we show that our model classifies two different locomotion types, the amoeboid and a so-called fan-shaped type.

The Open Science approach to complement and support the above framework and model is explained in Chapter 6. At first, we clarify certain notions regarding Open Science and research reproducibility. We, then, introduce our software package `AmoePy` and we describe different key aspects regarding (1) accessibility and understandability, (2) installation and usability, (3) archiving and extensibility, and (4) functionality and stability. Furthermore, we discuss the importance of our Open Science approach for the cell migration community and compare it to existing Open Science contributions.

Finally, in Chapter 7, we present an outlook of our research and conclude with the key findings of this thesis.

Biophysical background

2

First, we provide a brief overview of the underlying model organism, *Dictyostelium discoideum*, and the physiological processes underlying amoeboid cell motility.

2.1 The model organism *Dictyostelium discoideum*

In this section, we describe the organism *Dictyostelium discoideum* in general and its life cycle in particular. Afterwards, we explain the purpose of model organisms and why *Dictyostelium discoideum* fulfills the definition of such a model organism. This section is mainly based on [9, 57, 58].

***Dictyostelium discoideum* and its life cycle.** The eukaryote *D. discoideum*, commonly referred to as slime mold, is a member of the taxonomic group *Amoebozoa* and is commonly found in deciduous forest soils with a natural habitat ranging from the Arctic to the tropics [57, 59]. Its diet consists of yeast cells and bacteria, which are ingested via phagocytosis, i.e., by engulfment and absorption of the other cell [57, 60]. However, axenic strain (Ax-2) cells of *D. discoideum* are capable of pinocytosing external nutrients through the cell membrane (so-called "cellular drinking") [61]. By allowing cells to perform pinocytosis of a nutrient solution, instead of feeding on bacteria, the cell cultivation becomes much easier.

In Fig 2.1 the life cycle of *D. discoideum* is shown. It begins with germinating spores from which unicellular amoebas emerge. During the first phase, called proliferation or growth phase, the amoebas (depicted in light-green) feed on other microscopic organisms, e.g., bacteria such as *E. coli* (depicted in blue), and replicate themselves by cell division (mitosis). This proliferation phase lasts approximately three to eight hours [9]. Under starvation conditions, i.e. after a depletion of nutrients, the second phase begins which is called the development phase [9]. Here, cyclic adenosine monophosphate (cAMP) triggers the aggregation to a single migrating slug, which contains tens of thousands amoebas [57]. This transition from unicellular amoebas to a single multicellular organism is distinctive for this organism. In Fig 2.1, it is highlighted by a color change from light-green to dark-green. During the development phase, the slug migrates towards light sources and along temperature gradients [9]. Furthermore, a differentiation of cells takes place with dead cells building a thin stalk (depicted in brown) and remaining living cells forming the fruiting body (depicted in dark-green) of the slime mold. In the final step, new spores (depicted in light-green) are dispersed from the fruiting body starting a new life cycle of *D. discoideum* cells after approximately 24 hours. For more details on the the life cycle, see [9].

Rationale behind the terminology "model organism". Model organisms are non-human species to study a wide range of biological processes and phenomena, for

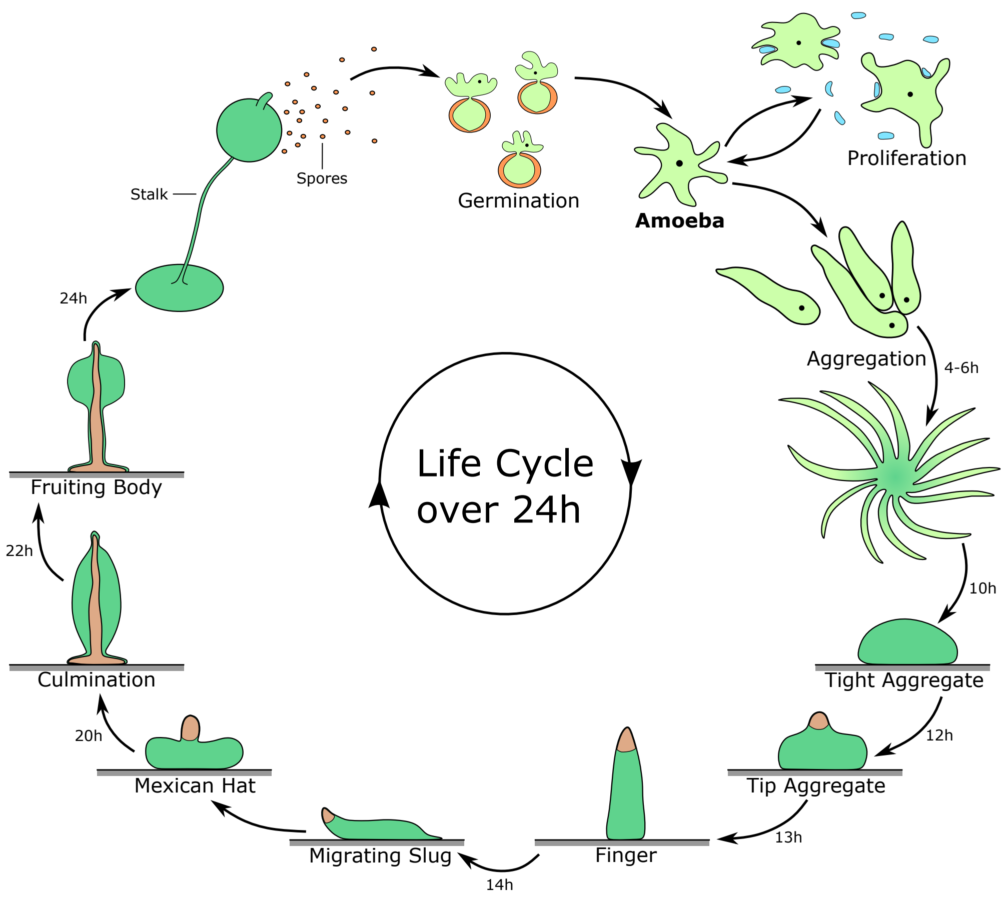


Figure 2.1. **Life cycle of *Dictyostelium discoideum*.** Unicellular amoebas are depicted as light-green contours with a black dot representing the cell nucleus. After a depletion of nutrients (depicted in blue), the development phase begins, in which a collection of amoebas aggregates to a multicellular slug (dark-green). The part of the slug which evolves into the fruiting body is depicted in dark-green, while the part which creates the stalk is depicted in brown. This figure is based on [62], [63, p. 5], and [64, p. 18].

which the inferred knowledge is then applied to other organisms [58]. While model organisms account only for a small fraction of all living species, the majority of biological insights is derived from the research and studies focusing on these organisms [65]. The broad and interdisciplinary use of model organisms is a result of (1) useful intrinsic characteristics, e.g., a simple and inexpensive culturation and handling, a small genome, and a high mutability and (2) further characteristics attributed to the organisms by the research community, e.g., a potential representation of a large number of other organisms and the ability to comprise multiple physiological processes at once, see [58] for more details. Furthermore, methods and experimental settings to study model organisms are, in general, well documented and, oftentimes, openly accessible along the underlying research data. For this reason, model organisms play an important role for the Open Science community. Examples of model organisms include: the bacterium *Escherichia coli*, the fruit fly

Drosophila melanogaster, the zebra fish *Danio rerio*, the budding yeast *Saccharomyces cerevisiae*, the weed Arabidopsis thaliana, and the house mouse *Mus musculus* [58].

Regarding *D. discoideum*, the cultivation is done in relatively simple and inexpensive. Due to its short life cycle of approximately $24h$, every stage of the cultivation process can be swiftly realized. Furthermore, spores and amoebas of different strains can be stored easily, cheaply, and indefinitely via freezing and thawing [8, 9]. By focusing on *D. discoideum*, different physiological processes can be studied, e.g., amoeboid cell motility, chemotaxis, cell adhesion, mytosis, phagocytosis, micropinocytosis, cell formation, and cell differentiation. Finally, the genetic malleability of *D. discoideum* enables to "(1) disrupt, replace, or silence a gene; (2) recover a mutated unknown gene; (3) and overexpress and/or tag a gene" ([8], as cited in [9]). Hence, *D. discoideum* meets the criteria of a model organism.

2.2 Amoeboid cell motility

Amoeboid movement belongs to the most widespread kinds of eukaryotic cell motility [66, 67]. It plays a key role in many biophysical and physiological processes such as wound healing, cancer metastasis, and immune system responses and is characterized by dynamic changes of the cell shape [2, 3, 4]. In this context, the cell is moving forward by creating protrusions, so-called pseudopodia. The location of these protrusions and the frequency of their formations mainly define the overall trajectory of the cell [68]. In addition to these explorative and fast-acting membrane protrusions, the cell retracts its rear (so-called uropod) in a slower and steadier way. By this coordinated interplay of protrusions and retractions, the cell can move persistently and efficiently in a crawling-like fashion. Finally, the cell track is affected by the creation and rupture of adhesion contacts to the substrate [69].

On an intracellular level, amoeboid migration is initialized by changes of the actin cytoskeleton, a microfilament meshwork consisting of many multi-functional proteins. More precisely, the cytoskeleton grows by polymerization and shrinks by depolymerization of actin filaments. This rearrangement of the actin network is controlled by additional cytoskeletal proteins initializing capping, severing, and nucleation of actin filaments creating different meshwork formations such as bundling, cross-linking, and branching [70]. The underlying mechanics are regulated by biochemical signaling pathways initializing different actions such as proliferation by cell growth and cell division, cell aggregation, and coordinated cell death [71, 72]. Furthermore, signaling pathways are linked to membrane receptors, enabling the cell to move persistently and directly towards a chemical source (chemotaxis) by sensing gradients and extracellular cues [73].

The formation of pseudopodia is the key step of amoeboid motility. The size and shape of these pseudopodia vary significantly for different cell types [74]. Some organisms such as *Dictyostelium discoideum* even possess different modes of locomotion and can switch back and forth between them [75]. Therefore, pseudopodia are subdivided, based on their nature, into different types: lobopodia, lamellipodia,

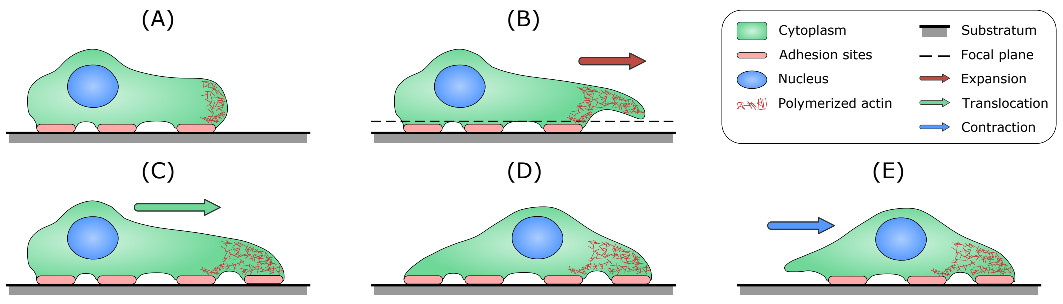


Figure 2.2. Amoeboid cell motility. First, polymerized actin initializes the forward movement of the cell (A) which results in an expansion at the leading edge, a so-called pseudopodium (B). After a new adhesion to the substratum is built (C), the translocation of the cell nucleus takes place (D). In the final phase, the rear edge of the cell is contracted (E). Additionally, in panel (B), the focal plain is shown as dashed gray line. This figure is based on [81].

filopodia, reticulopodia, axopodia, invadopodia, and others [76, 77, 78]. Lobopodia are characterized by cylindrical and finger-shaped protrusions, while lamellipodia possess a flat and broader structure [74]. On the contrary, filopodia are thinner and more elongated and consist mostly of ectoplasm [76]. Reticulopodia are characterized by an irregular and dense network of fine string-like extensions which interact with each other [78]. Axopodia are defined by thin and straight lines emerging in a radial way from the cell body [77]. In this work, we focus on lobopodia and lamellipodia. Pseudopodia can be also classified depending on the exact location of their appearance: Y-shaped (split) pseudopodia at the front, and *de novo* pseudopodia at the left, right, and rear side of the cell changing the direction of its movement [6]. Moreover, one distinguishes lobopodia from so-called blebs, a type of smaller and fast-paced protrusions, which are initialized by the internal pressure of the cell and rupture of its membrane [79, 80].

In Fig 2.2, a schematic overview of the different stages during amoeoboid cell motility is shown. By polymerization of monomeric actin, so-called globular actin (G-actin), a meshwork of filamentous actin (F-actin) is formed at the cell's front (panel (A)). While F-actin can be found throughout the cell body, its density is increased at the cell's front, leading to a membrane expansion and a forward movement of the cell (panel (B)). By retraction of the leading edge, which is controlled by actin depolymerization, the cell can move back to the initial stage. If the expanding membrane builds a new adhesion site (panel (C)), the nucleus of the cell is pulled towards the leading edge (panel (D)). Finally, the cell's rear detaches from the substratum, which is followed by a retraction of the trailing edge towards the cell's center (panel (E)). In panel (B), an exemplary focal plane of the recorded microscopy images is displayed, indicating the following experimental challenges: (1) cell parts which are not in focus and therefore missed by the imaging process and (2) membrane expansions dropping into the focal plane which are then displayed as isolated areas visually separated from the main cell body in the 2D image frame.

Mathematical background 3

For a better understanding, we provide a mathematical overview of the underlying mathematical concepts used in Chapter 4 and 5. In Section 3.1, we introduce the Gaussian process regression (GPR), its usage to obtain estimated cell outlines (so-called cell contours), and further information regarding the computation of the contour curvature and the estimation of the underlying hyperparameters. Then, in Section 3.2, we introduce geometric contour flows, e.g., the curve-shortening flow (CSF) and the area-preserving curve-shortening flow (APCSF), used in our amoeboid cell motility model described later on in Chapter 5. Finally, we provide further information on the two stochastic processes used in our model: the Hawkes process (HP) and the Ornstein-Uhlenbeck process (OUP).

3.1 Gaussian process regression

In this section, we characterize briefly the main aspects of the Gaussian process regression (GPR) which we used to determine smooth two-dimensional representations of cell membranes. In this context, we introduce common notations and formulas which are used in the following sections. In general, the GPR, also known as "kriging", offers a Bayesian framework for regression analysis. It does not only provide a single regression function but also quantifies the associated uncertainty. Detailed studies of Gaussian processes in machine learning and especially in regression analysis can be found in many textbooks such as [82, 83]. In [84, 85] short summaries of the GPR are presented. Details about the implementation of the GPR can be found in [86].

3.1.1 General definitions and theory

First, we assume a training data set $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \dots, n\}$ consisting of n observations. The input vectors are denoted as $\mathbf{x}_i \in \mathbb{R}^d$, $d \in \mathbb{N}$ and the corresponding scalar outputs are denoted as $y_i \in \mathbb{R}$. Now, we are interested in a distribution of regression functions $f : \mathbb{R}^d \to \mathbb{R}$ which are described by a Gaussian process (GP). Due to the fact that a GP is uniquely defined by its mean function $\mathbf{x} \mapsto \mu(\mathbf{x})$ and its covariance function $(\mathbf{x}, \mathbf{x}') \mapsto \Sigma(\mathbf{x}, \mathbf{x}')$, we introduce the following notation:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}),\ \Sigma(\mathbf{x}, \mathbf{x}')\right),$$

where the mean function and the covariance function are given by

$$\begin{aligned} \mu(\mathbf{x}) &= \mathbb{E}\left[f(\mathbf{x})\right], \\ \Sigma(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))\right]. \end{aligned}$$

Since we assumed n observations obtained by d-dimensional input vectors, we define the training input as $X \in \mathbb{R}^{d\times n}$, which is also called design matrix. The training output for all n observations are denoted as $Y \in \mathbb{R}^n$. Similarly, we denote the testing input and output for $n_* \in \mathbb{N}$ evaluations as $X_* \in \mathbb{R}^{d\times n_*}$ and $Y_* \in \mathbb{R}^{n_*}$, respectively.

Prediction model with noise. For the expected values of Y and Y_* we introduce the following notation:

$$\mu := \mathbb{E}[Y] \quad \text{and} \quad \mu_* := \mathbb{E}[Y_*].$$

We assume noisy observations with noise variance σ^2_{noise} such that the covariance of Y is given as

$$\operatorname{Cov}(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma^2_{\text{noise}}\delta_{ij} \quad \text{and} \quad \operatorname{Cov}(Y, Y) = K(X, X) + \sigma^2_{\text{noise}} I_n, \tag{3.1}$$

where the kernel matrix K is defined as

$$K(X, X') := \left(k(\mathbf{x}_i, \mathbf{x}'_j)\right)_{i,j},$$

for a given kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, e.g., the Gaussian kernel function

$$k_\delta(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\delta^2}\right), \qquad \delta > 0.$$

Likewise, we define the following covariance matrices:

$$\begin{aligned} \operatorname{Cov}(Y, Y_*) = \operatorname{Cov}(Y_*, Y)^{\mathsf{T}} &= K(X, X_*), \\ \operatorname{Cov}(Y_*, Y_*) &= K(X_*, X_*). \end{aligned} \tag{3.2}$$

Finally, we assume that the training outputs Y and test outputs Y_* are jointly normally distributed:

$$\begin{pmatrix} Y \\ Y_* \end{pmatrix} \sim \mathcal{N}_{n+n_*}\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{bmatrix} K(X, X) + \sigma^2_{\text{noise}} I_n & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right).$$

By using the abbreviated notation Σ, Σ_* and Σ_{**}, we obtain:

$$\begin{pmatrix} Y \\ Y_* \end{pmatrix} \sim \mathcal{N}_{n+n_*}\left(\begin{pmatrix} \mu \\ \mu_* \end{pmatrix}, \begin{bmatrix} \Sigma & \Sigma_* \\ \Sigma_*^{\mathsf{T}} & \Sigma_{**} \end{bmatrix}\right).$$

Since we are interested in inferring the relationship between inputs and function evaluations, the conditional distribution of the test output Y_* given the training output Y with underlying inputs X and X_* is of relevance. This conditional distribution is also normally distributed

$$Y_* \mid Y \sim \mathcal{N}_{n_*}\left(\tilde{\mu}, \tilde{\Sigma}\right), \tag{3.3}$$

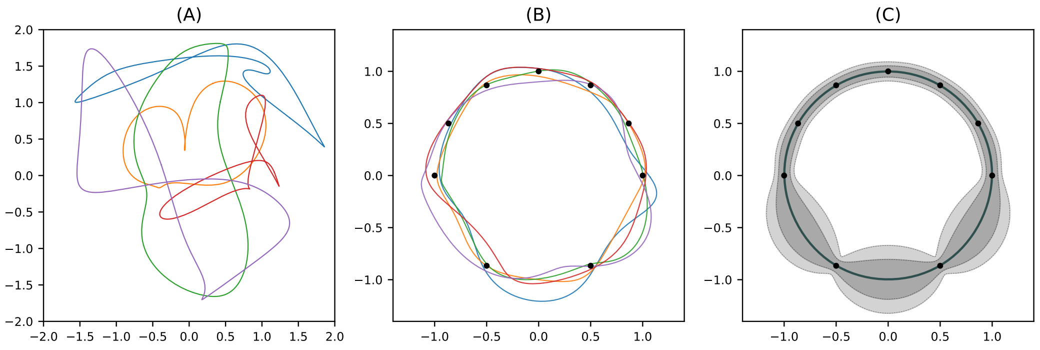


Figure 3.1. GPR with samples drawn from GP prior and GP posterior. (A) Functions (colored lines) sampled from the GP prior distribution from Eq (3.5). **(B)** Based on training data (black dots), functions are now sampled w.r.t to the GP posterior distribution from Eq (3.3). **(C)** GPR fit (dark gray line) with corresponding credible intervals: 75% (gray) and 95% (light gray). As underlying kernel function a Poisson kernel from Eq (3.11) was used.

with mean and covariance defined as

$$\begin{aligned}\tilde{\mu} &= \mathbb{E}(Y_* \mid Y) = \mu_* + \Sigma_*^{\mathsf{T}} \Sigma^{-1} (Y - \mu) \\ \tilde{\Sigma} &= \mathbb{V}(Y_* \mid Y) = \Sigma_{**} - \Sigma_*^{\mathsf{T}} \Sigma^{-1} \Sigma_* .\end{aligned} \tag{3.4}$$

The distribution from Eq (3.3) is called the GP posterior distribution, whereas the GP prior distribution is given by

$$Y_* \sim \mathcal{N}_{n_*} (\mu_*, \Sigma_{**}) . \tag{3.5}$$

However, in most cases, it is not necessary to sample from these distributions, since a regression function is already provided by the (analytic) expectation $\mathbb{E}(Y_* \mid Y) = \mu_* + \Sigma_*^{\mathsf{T}} \Sigma^{-1} (Y - \mu)$. In general, this function is meant when we speak of “Gaussian process regression”.

In Fig 3.1, five samples, drawn from the GP prior and the GP posterior, are displayed in panels (A) and (B), respectively. The samples for the GP prior are periodic but very chaotic. Even for a small number of training data points, the GP posterior produces already reasonable fits (panel (B)). Finally, the GPR fit based on $\tilde{\mu}$ from Eq (3.4) is displayed in panel (C). From the credible intervals (75% and 95%), we observe the impact of the training data points on the regression function and, especially, the samples drawn from the GP posterior. The credible intervals are more narrow in the upper half of panel (C), due to an increased number of data points located in close proximity to each other.

Choice of hyperparameters via maximized likelihood. Consider the normally distributed observations $Y \sim \mathcal{N}_n(\mu(\vartheta), \Sigma(\vartheta))$ depending on parameters $\vartheta \in \Theta \subset \mathbb{R}^{n_\theta}, n_\theta \in \mathbb{N}$. To estimate the parameters, we maximize the likelihood function

$L_{\mathbf{y}} : \Theta \to \mathbb{R}_0^+$, $L_{\mathbf{y}}(\vartheta) := p_\vartheta(\mathbf{y})$. The density function of the normally distributed random variable Y is given by:

$$p_\vartheta(\mathbf{y}) := \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} \exp\left(-\frac{1}{2}\left(\mathbf{y}-\mu\right)^\mathsf{T}\Sigma^{-1}\left(\mathbf{y}-\mu\right)\right),$$

where $\mu := \mu(\vartheta)$ and $\Sigma := \Sigma(\vartheta)$. The log-likelihood is then given by

$$\ell_{\mathbf{y}}(\vartheta) := \log L_{\mathbf{y}}(\vartheta) = -\frac{1}{2}\log|\Sigma| - \frac{1}{2}(\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}(\mathbf{y}-\mu) - \frac{n}{2}\log(2\pi). \tag{3.6}$$

Partial derivatives of $\ell_{\mathbf{y}}(\vartheta)$. Assuming separate parameters ϑ_μ and ϑ_Σ impacting the expectation $\mu := \mu(\vartheta_\mu)$ and the covariance $\Sigma := \Sigma(\vartheta_\Sigma)$, the following partial derivatives of the log-likelihood function from Eq (3.6) can be derived:

$$\begin{aligned}\frac{\partial \ell_{\mathbf{y}}}{\partial \vartheta_\mu} &= (\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}\frac{\partial \mu}{\partial \vartheta_\mu} \quad \text{and} \\ \frac{\partial \ell_{\mathbf{y}}}{\partial \vartheta_\Sigma} &= -\frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\frac{\partial \Sigma}{\partial \vartheta_\Sigma}\right) + \frac{1}{2}(\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}\frac{\partial \Sigma}{\partial \vartheta_\Sigma}\Sigma^{-1}(\mathbf{y}-\mu).\end{aligned} \tag{3.7}$$

This can be seen as follows. At first, by using basic mathematical operations, we can show that:

$$\begin{aligned}\frac{\partial \ell_{\mathbf{y}}}{\partial \vartheta_\mu} &= 0 + \frac{\partial}{\partial \vartheta_\mu}\left(-\frac{1}{2}(\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}(\mathbf{y}-\mu)\right) + 0 \\ &= \frac{1}{2}\frac{\partial \mu}{\partial \vartheta_\mu}^\mathsf{T}\Sigma^{-1}(\mathbf{y}-\mu) + \frac{1}{2}(\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}\frac{\partial \mu}{\partial \vartheta_\mu} \\ &= \frac{1}{2}\left(\frac{\partial \mu}{\partial \vartheta_\mu}^\mathsf{T}\Sigma^{-1}(\mathbf{y}-\mu)\right)^\mathsf{T} + \frac{1}{2}(\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}\frac{\partial \mu}{\partial \vartheta_\mu} \\ &= \frac{1}{2}(\mathbf{y}-\mu)^\mathsf{T}\left(\Sigma^{-1}\right)^\mathsf{T}\frac{\partial \mu}{\partial \vartheta_\mu} + \frac{1}{2}(\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}\frac{\partial \mu}{\partial \vartheta_\mu} \\ &\overset{1}{=} (\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}\frac{\partial \mu}{\partial \vartheta_\mu}.\end{aligned}$$

Now, we consider the following identities of matrix derivatives:

$$\frac{dA^{-1}}{d\vartheta} = -A^{-1}\frac{dA}{d\vartheta}A^{-1} \quad \text{and} \quad \frac{d\,|A|}{d\vartheta} = |A|\,\mathrm{tr}\left(A^{-1}\frac{dA}{d\vartheta}\right), \tag{3.8}$$

where $A = A(\vartheta)$ is an arbitrary matrix depending on a parameter ϑ. By using these identities, it is easy to show that:

$$\begin{aligned}\frac{\partial \ell_{\mathbf{y}}}{\partial \vartheta_\Sigma} &= -\frac{1}{2}\frac{\partial}{\partial \vartheta_\Sigma}\left(\log|\Sigma|\right) - \frac{1}{2}\frac{\partial}{\partial \vartheta_\Sigma}\left((\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}(\mathbf{y}-\mu)\right) \\ &= -\frac{1}{2\,|\Sigma|}|\Sigma|\,\mathrm{tr}\left(\Sigma^{-1}\frac{\partial \Sigma}{\partial \vartheta_\Sigma}\right) - \frac{1}{2}\left(-(\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}\frac{\partial \Sigma}{\partial \vartheta_\Sigma}\Sigma^{-1}(\mathbf{y}-\mu)\right) \\ &= -\frac{1}{2}\mathrm{tr}\left(\Sigma^{-1}\frac{\partial \Sigma}{\partial \vartheta_\Sigma}\right) + \frac{1}{2}(\mathbf{y}-\mu)^\mathsf{T}\Sigma^{-1}\frac{\partial \Sigma}{\partial \vartheta_\Sigma}\Sigma^{-1}(\mathbf{y}-\mu).\end{aligned}$$

[1] $\left(\Sigma^{-1}\right)^\mathsf{T} = \left(\Sigma^\mathsf{T}\right)^{-1} = \Sigma^{-1}$ due to the symmetry of Σ.

3.1.2 Interpolation model of cell outlines

In this section, we describe how to use the GPR to interpolate cell outlines based on a discrete set of segmentation points. We consider only a single contour Γ_k; for ease of notation, we omit the subscript k in the sequel. Let $(x_0, y_0), \dots, (x_{M-1}, y_{M-1})$ denote $M \in \mathbb{N}$ segmentation points with periodic boundaries $x_M \coloneqq x_0$ and $y_M \coloneqq y_0$ since we describe cell contours by closed curves. In the following, when we speak of curves, we always refer to closed simple curves, i.e., curves without self-intersections.

The idea is to use the arc length of these curves as parametrization to obtain corresponding coordinates in $\mathbb{R}^2$. This parametrization of the contour Γ will be denoted as

$$\Phi : [0, 2\pi) \ni \theta \mapsto (x(\theta), y(\theta)) \in \mathbb{R}^2 \tag{3.9}$$

First, we define support points $\theta_i,\ i \in \{0, 1, \dots, M-1\}$

$$\theta_m = \frac{2\pi \sum_{i=0}^{m} d_i}{\sum_{i=0}^{M-1} d_i} \quad \text{and} \quad d_0 = 0 \tag{3.10}$$

according to the normalized secant length along the contour

$$d_i = \left((x_i - x_{i-1})^2 + (y_i - y_{i-1})^2\right)^{1/2}$$

It is easy to see that the sequence of support points obtained by Eq (3.10) is strictly increasing, i.e. $\theta_0 < \theta_1 < \dots < \theta_{M-1}$.

Now, we apply the GPR to this sequence of support points in order to obtain a smoothing spline representing the cell contour. The underlying kernel function, necessary to model the correlation between the data points, is defined by

$$k_r(\theta, \theta') \coloneqq \frac{1 - r^2}{1 - 2r\ \cos(\theta - \theta') + r^2}, \qquad \theta, \theta' \in [0, 2\pi), \tag{3.11}$$

with radius parameter $r \in (0, 1)$. This kernel function, which is closely related to the wrapped Cauchy distribution, is often called **Poisson kernel function**. We have chosen the Poisson kernel due to its periodicity property, i.e., $k_r(\theta, \theta') = k_r(\theta + 2z\pi, \theta')$ for all $z \in \mathbb{Z}$. Alternatively, one could periodize a non-periodic kernel function $k(\cdot, \cdot)$, e.g., the Gaussian kernel function, by applying the following formula:

$$k_{\text{per}}(\theta, \theta') \coloneqq \sum_{z \in \mathbb{Z}} k(\theta + 2z\pi, \theta').$$

We simplify the regression model by assuming that x and y coordinates are independent from each other. For this reason, we perform the GPR two times in order to obtain two regression functions f_x and f_y satisfying $f_x(\theta_i) \approx x_i$ and $f_y(\theta_i) \approx y_i$ for all $i \in \{0, \dots, M-1\}$. By deducing these regression functions, we can compute the two-dimensional coordinates for any test input $\tilde{\theta}_j \in [0, 2\pi)$ with $j \in \{0, \dots, M_* - 1\}$ and $M_* \in \mathbb{N}$ representing the number of function evaluations.

In the following, we describe how to obtain the x coordinates $\tilde{\mathbf{x}} := (\tilde{x}_0, \dots, \tilde{x}_{M_*-1})$ given the training input $\{\theta_j \mid j = 0, \dots, M-1\}$, the training output $\mathbf{x} := (x_0, \dots, x_{M-1})$ and the testing input $\{\tilde{\theta}_j \mid j = 0, \dots, M_* - 1\}$. The GPR is then similarly used to obtain y coordinates of the curve. First, we consider a prediction model with noisy observations and no drift, i.e. $\mathbb{E}[\mathbf{x}] = \mathbb{E}[\tilde{\mathbf{x}}] = 0$. The covariances are given by the Eq (3.1) and (3.2), where the underlying kernel matrix $K(\boldsymbol{\theta}, \boldsymbol{\theta}') := \left(k(\theta_i, \theta'_j)\right)_{i,j}$ is given by the Poisson kernel from Eq (3.11).

By assuming centered Gaussian processes, the predictive equations of the GPR are simply given by:

$$\begin{aligned} \mathbb{E}(\tilde{\mathbf{x}} \mid \mathbf{x}) &= \Sigma_*^{\mathsf{T}} \Sigma^{-1} \mathbf{x} \qquad \text{and} \\ \mathbb{V}(\tilde{\mathbf{x}} \mid \mathbf{x}) &= \Sigma_{**} - \Sigma_*^{\mathsf{T}} \Sigma^{-1} \Sigma_*. \end{aligned} \tag{3.12}$$

Since the covariance matrices depend on the input data consisting of the normalized arc length parametrization, we introduce the following notation:

$$\begin{aligned} f_x(\tilde{\boldsymbol{\theta}}) &:= \mathbb{E}(\tilde{\mathbf{x}} \mid \mathbf{x}; \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \cdot \Sigma^{-1}(\boldsymbol{\theta}) \cdot \mathbf{x}, \\ f_y(\tilde{\boldsymbol{\theta}}) &:= \mathbb{E}(\tilde{\mathbf{y}} \mid \mathbf{y}; \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \cdot \Sigma^{-1}(\boldsymbol{\theta}) \cdot \mathbf{y}. \end{aligned} \tag{3.13}$$

As in Fig 3.1, the predictive equations from Eq (3.12) provide a function space of infinitely many smoothing splines; each drawn with the GP posterior distribution based on $\mathbb{E}(\tilde{\mathbf{x}} \mid \mathbf{x})$ and $\mathbb{V}(\tilde{\mathbf{x}} \mid \mathbf{x})$. However, the regression functions $f_x : [0, 2\pi)^{M_*} \to \mathbb{R}^{M_*}$ and $f_y : [0, 2\pi)^{M_*} \to \mathbb{R}^{M_*}$ are simply defined by the expectations $\mathbb{E}(\tilde{\mathbf{x}} \mid \mathbf{x}; \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta})$ and $\mathbb{E}(\tilde{\mathbf{y}} \mid \mathbf{y}; \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta})$, which can be computed analytically.

3.1.3 Estimation of curvature using GPR

In this section, we highlight the benefits of using the GPR to obtain smooth cell outlines given a discrete set of segmentation points. For any simply closed curve in a two-dimensional space

$$\begin{aligned} \gamma : [0, 2\pi) &\to \mathbb{R}^2, \\ \theta \mapsto \gamma(\theta) &:= \begin{pmatrix} f_x(\theta) \\ f_y(\theta) \end{pmatrix} \end{aligned}$$

with corresponding coordinate functions $f_x : [0, 2\pi) \to \mathbb{R}$ and $f_y : [0, 2\pi) \to \mathbb{R}$, the curvature at $\gamma(\theta)$ is denoted as $\kappa(\theta)$ and defined by

$$\kappa(\theta) := \frac{f'_x f''_y - f'_y f''_x}{\left(f'^2_x + f'^2_y\right)^{\frac{3}{2}}}, \qquad \theta \in [0, 2\pi). \tag{3.14}$$

By applying the GPR, described as in Eq (3.13), we obtain coordinates of the estimated curve. Additionally, we can easily compute the first and second derivative of f_x:

$$f_x'(\tilde{\boldsymbol{\theta}}) = \frac{\partial}{\partial \tilde{\boldsymbol{\theta}}} \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \cdot \Sigma^{-1}(\boldsymbol{\theta}) \cdot \mathbf{x} \quad \text{and} \quad f_x''(\tilde{\boldsymbol{\theta}}) = \frac{\partial^2}{\partial \tilde{\boldsymbol{\theta}}^2} \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \cdot \Sigma^{-1}(\boldsymbol{\theta}) \cdot \mathbf{x}, \tag{3.15}$$

and similarly for f_y:

$$f_y'(\tilde{\boldsymbol{\theta}}) = \frac{\partial}{\partial \tilde{\boldsymbol{\theta}}} \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \cdot \Sigma^{-1}(\boldsymbol{\theta}) \cdot \mathbf{y} \quad \text{and} \quad f_y''(\tilde{\boldsymbol{\theta}}) = \frac{\partial^2}{\partial \tilde{\boldsymbol{\theta}}^2} \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) \cdot \Sigma^{-1}(\boldsymbol{\theta}) \cdot \mathbf{y}. \tag{3.16}$$

The partial derivative $\frac{\partial}{\partial \tilde{\boldsymbol{\theta}}} \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}})$ is given by the partial derivative of the Poisson kernel function k_r. More precisely, we obtain the following equations:

$$\begin{aligned} \frac{\partial}{\partial \tilde{\boldsymbol{\theta}}} \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) &= \frac{\partial}{\partial \tilde{\boldsymbol{\theta}}} K(\tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \left(\frac{\partial}{\partial \tilde{\theta}_i} k_r(\tilde{\theta}_i, \theta_j) \right)_{i,j} \qquad \text{and} \\ \frac{\partial^2}{\partial \tilde{\boldsymbol{\theta}}^2} \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) &= \frac{\partial^2}{\partial \tilde{\boldsymbol{\theta}}^2} K(\tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \left(\frac{\partial^2}{\partial \tilde{\theta}_i^2} k_r(\tilde{\theta}_i, \theta_j) \right)_{i,j} . \end{aligned}$$

For the Poisson kernel function in Eq (3.11), the first and second partial derivatives are given by

$$\frac{\partial}{\partial \tilde{\theta}} k_r(\tilde{\theta}, \theta) = \frac{2r\,(r^2-1)\sin(\tilde{\theta}-\theta)}{\left(r^2 - 2r\ \cos(\tilde{\theta}-\theta) + 1\right)^2}$$

and

$$\frac{\partial^2}{\partial \tilde{\theta}^2} k_r(\tilde{\theta}, \theta) = \frac{2r(r^2-1)\Big((r^2+1)\cos(\tilde{\theta}-\theta) - 2r\ \cos^2(\tilde{\theta}-\theta) - 4r\ \sin^2(\tilde{\theta}-\theta)\Big)}{\left(r^2 - 2r\ \cos(\tilde{\theta}-\theta) + 1\right)^3}.$$

In order to compute f_x' and f_x'', we have to substitute the first covariance matrix in Eq (3.13) with the covariance matrix generated by the first and second derivative of the kernel function, respectively. Likewise to f_x, we can evaluate the derivatives f_x' and f_x'' at every position $\tilde{\theta}$. The same holds true for f_y, f_y', and f_y''. Thereby we found an analytic way to compute the curvature κ via the GPR predictions from Eq (3.14).

3.1.4 Parameter estimation

In this section, we present a short overview of how to estimate the hyperparameters r and σ_{noise} underlying the GPR to obtain smooth cell contours. A common approach is the maximum likelihood estimation (MLE) method, where the set of hyperparameters is chosen such that the corresponding log-likelihood function from Eq (3.6) is maximized. While this approach is very popular, potential issues have been reported including the inefficiency for problems of higher complexity, higher computational costs, and the possibility of ill-posed estimation problems [87, 88, 89]. For this reason, different strategies have been proposed to improve the MLE method for GPR [87] and alternative methods have been tested successfully, e.g., cross-validation [90, 91]. Nevertheless, we decided to follow the standard MLE approach due to the minor complexity of the resulting regression functions, i.e., one-dimensional closed curves in a two-dimensional domain, and only two hyperparameters to be estimated.

We remind that the GPR model is based on centered Gaussian processes (i.e. $\mu = 0$) and that both hyperparameters influence the covariance matrix Σ only. Hence, for the partial derivative of the log-likelihood function from Eq (3.6), we obtain the following formula

$$\frac{\partial \ell_{\mathbf{y}}}{\partial \vartheta_\Sigma} = -\frac{1}{2}\operatorname{tr}\left(\Sigma^{-1}\frac{\partial \Sigma}{\partial \vartheta_\Sigma}\right) + \frac{1}{2}\mathbf{y}^\mathsf{T}\Sigma^{-1}\frac{\partial \Sigma}{\partial \vartheta_\Sigma}\Sigma^{-1}\mathbf{y} \tag{3.17}$$

with $\vartheta_\Sigma \in \{r, \sigma_{\text{noise}}\}$. Furthermore, $\partial\Sigma/\partial\vartheta_\Sigma$ is given by the following formulas

$$\frac{\partial \Sigma}{\partial r} = \frac{\partial}{\partial r}\left(K(\boldsymbol{\theta},\boldsymbol{\theta}) + \sigma_{\text{noise}}^2 I_n\right) = \left(\frac{\partial}{\partial r}k_r(\theta_i,\theta_j)\right)_{i,j} \qquad \text{and}$$

$$\frac{\partial \Sigma}{\partial \sigma_{\text{noise}}} = \frac{\partial}{\partial \sigma_{\text{noise}}}\left(K(\boldsymbol{\theta},\boldsymbol{\theta}) + \sigma_{\text{noise}}^2 I_n\right) = 2\sigma_{\text{noise}} I_n$$

with the corresponding derivative of the Poisson kernel function

$$\frac{\partial k_r(\theta,\theta')}{\partial r} = \frac{2(r^2+1)\cos(\theta-\theta') - 4r}{(1-2r\ cos(\theta-\theta')+r^2)^2}.$$

Now, we can determine the maximum log-likelihood estimate using gradient descent with the above derivatives in order to estimate the hyperparameters. In Fig 3.2, we present estimated parameters based on a persistently motile cell example. In panel (A), pairs of estimated hyperparameters $(r, \sigma_{\text{noise}})$ are shown for different contours $(K = 500)$. The gradient descent method was applied to each contour separately in order to obtain one pair of hyperparameters.

Two clusters can be observed, which corresponds to the bimodal distributions of r and σ_{noise} shown in panel (B) and (C). Furthermore, the total log-likelihood, i.e., the sum of the log-likelihood functions of all contours, is presented in panel (D). The maximum at $r = 0.80$ and $\sigma_{\text{noise}} = 0.017$ is depicted as a black dot. These parameters are used in the following for all computations in Chapter 4.

3.1.5 The "amoeba tube" representation and corresponding surface curvatures

In Section 3.1.3, we demonstrated how to compute the curvature of single cell contours by using the GPR. In this section, we present an approach to display the contour dynamics for the entire time span. By stacking consecutive contours onto each other, we obtain a 3D representation, which we call "amoeba tube". Furthermore, we show that the GPR can be used to obtain smooth predictions for these 3D objects. The initial idea was to compute the differential geometric properties

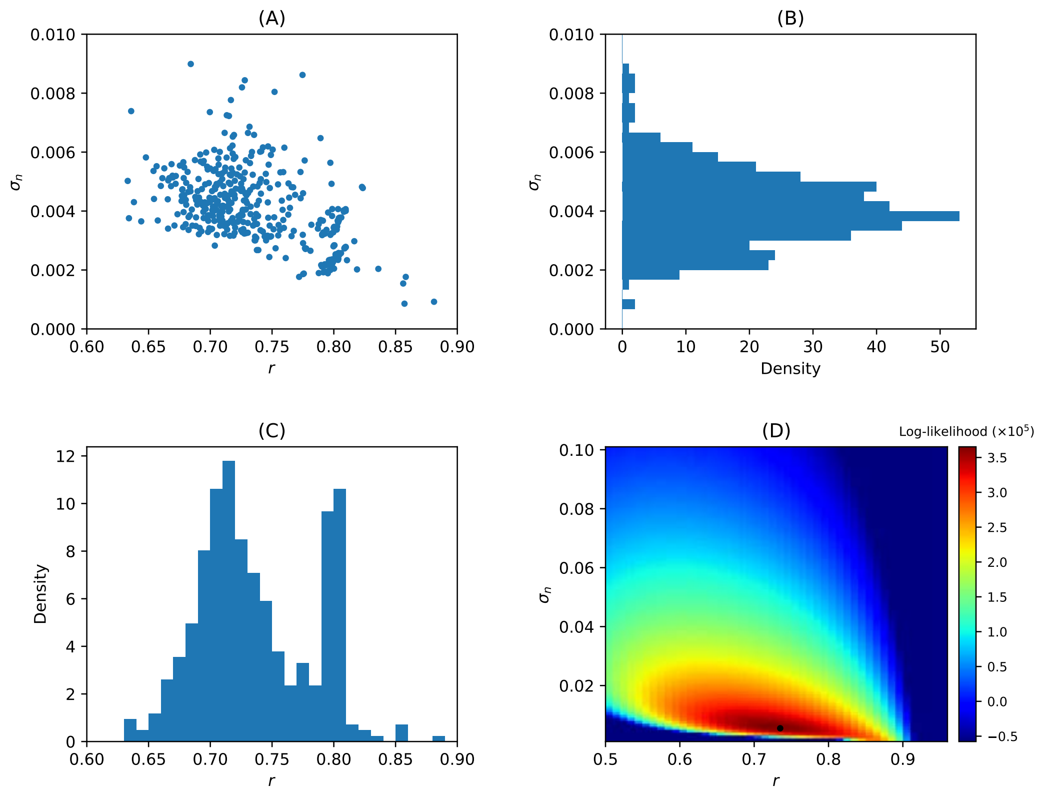


Figure 3.2. Estimation of hyperparameters for example cell track. (A) $K = 500$ Pairs of parameters $(r, \sigma_{\text{noise}})$ estimated by maximum likelihood for each contour. **(B, C)** Corresponding histograms for σ_{noise} and r, respectively. **(D)** Total log-likelihood of all $K = 500$ contours as a function of the hyperparameters. The maximum is displayed as a black dot.

of these smooth amoeba tubes to identify protrusions/retractions of the underlying contour dynamics. By assuming a geometric scaling of $1s \widehat{=} 1\mu m$, we use the GPR to compute different surface curvatures: the Gaussian curvature, the mean curvature, and the principal curvatures. In general, this approach faces some challenges regarding (1) the efficiency to detect/classify membrane protrusions/retractions and (2) the numerical feasibility for large data sets. Nevertheless, we want to show our findings since the underlying method might be useful for other applications. In Fig 3.3, we illustrate an amoeba tube of an experimental cell track. The x- and y-coordinates, which reflect the spatial position of each segmentation point, are displayed horizontally, while the vertical axis reflects the temporal coordinate of each contour/segmentation point.

In order to apply the GPR to predict a smooth amoeba tube, we introduce the following parametrization:

$$f : [0, 2\pi) \times \mathbb{R}^+ \to \mathbb{R}^3, \tag{3.18}$$

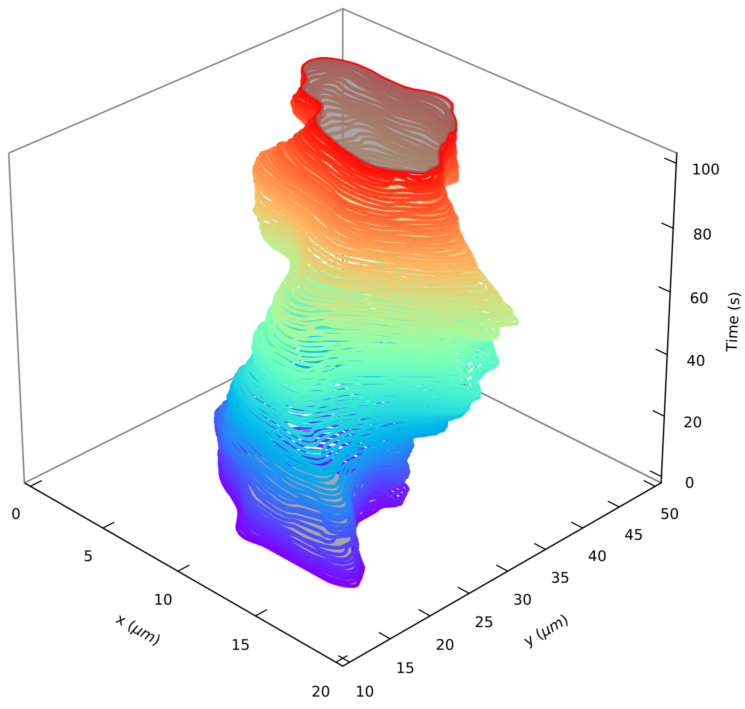


Figure 3.3. **Amoeba tube representation** Three-dimensional representation of experimental contour dynamics, where consecutive contours are stacked onto each other. The color of each contour coincides with the time scale.

$$(\theta, t) \mapsto f(\theta, t) := \begin{pmatrix} f_x(\theta, t) \\ f_y(\theta, t) \\ t \end{pmatrix}$$

for time $t \in \mathbb{R}^+$ and normalized arc length coordinate $\theta \in [0, 2\pi)$. As an alternative for more simple data sets, θ can be defined as polar coordinate, see [22, 92].

We will use a GPR model with correlation matrices based on two kernels: the Poisson kernel from Eq (3.11) and the following Gaussian Kernel:

$$k_{\sigma,\delta}(t, t') := \sigma^2 exp\left(\frac{-(t-t')^2}{2\delta^2}\right), \qquad \sigma, \delta > 0. \tag{3.19}$$

While the Poisson kernel defines the spatial correlation of our data set, the temporal correlation between contours is modeled with the Gaussian kernel. We assume a total number of n contours and that each contour is described by a sequence of m segmentation points. Hence, the covariance matrix Σ, containing the correlation information of the entire data set, i.e., every segmentation point of every contour, is of dimension $mn \times mn$. Each sequence of segmentation points is assumed to be equidistant, i.e., $\theta_0, \ldots, \theta_{m-1} \in [0, 2\pi)$ with $\theta_i = i \cdot 2\pi/m$, and, for now, somehow "sorted" with respect to the preceding contour/sequence of segmentation points. The exact details of the latter assumption are presented later on in Chapter 4.

Based on the two kernel functions mentioned above, we introduce the following spatial and temporal covariance matrices:

$$\begin{aligned}\Sigma_{\text{space}} &= (k_r(\theta_i, \theta_j))_{i,j<m}, \qquad && \theta_0 < \cdots < \theta_{m-1} \in [0, 2\pi)\\ \Sigma_{\text{time}} &= (k_{\sigma,\delta}(t_i, t_j))_{i,j<n}, && t_0 < \cdots < t_{n-1} \in [0, T].\end{aligned}$$

By taking the tensor product (or so-called Kronecker product) of these kernel functions, we now define the combined covariance matrix Σ by

$$\begin{aligned}\Sigma &= \Sigma_{\text{space}} \otimes \Sigma_{\text{time}} + \sigma_{\text{noise}}^2 I_{mn}\\ &= \begin{bmatrix} k_r(\theta_0,\theta_0)\Sigma_{\text{time}} & \dots & k_r(\theta_0,\theta_{m-1})\Sigma_{\text{time}} \\ \vdots & \ddots & \vdots \\ k_r(\theta_{m-1},\theta_0)\Sigma_{\text{time}} & \dots & k_r(\theta_{m-1},\theta_{m-1})\Sigma_{\text{time}} \end{bmatrix} + \sigma_{\text{noise}}^2 I_{mn},\end{aligned}$$

Analogous to (3.13), we apply the GPR to obtain the following regression functions:

$$\begin{aligned} f_x(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{t}}) &:= \mathbb{E}(\tilde{\mathbf{x}} \mid \mathbf{x}; \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}, \tilde{\mathbf{t}}, \mathbf{t}) = \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}, \tilde{\mathbf{t}}, \mathbf{t}) \cdot \Sigma^{-1}(\boldsymbol{\theta}, \mathbf{t}) \cdot \mathbf{x}, \\ f_y(\tilde{\boldsymbol{\theta}}, \tilde{\mathbf{t}}) &:= \mathbb{E}(\tilde{\mathbf{y}} \mid \mathbf{y}; \tilde{\boldsymbol{\theta}}, \boldsymbol{\theta}, \tilde{\mathbf{t}}, \mathbf{t}) = \Sigma_*^{\mathsf{T}}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}, \tilde{\mathbf{t}}, \mathbf{t}) \cdot \Sigma^{-1}(\boldsymbol{\theta}, \mathbf{t}) \cdot \mathbf{y}. \end{aligned} \tag{3.20}$$

with segmentation point coordinates $\mathbf{x}, \mathbf{y}, \mathbf{t} \in \mathbb{R}^{nm}$ and grid point coordinates $\tilde{\boldsymbol{\theta}} \in [0, 2\pi)^{k_*}, \tilde{\mathbf{t}} \in \mathbb{R}^{k_*}$ for k_* evaluations. Based on these regression functions, we can use f from Eq (3.18) to obtain the predicted position of the amoeba tube surface for any arc length coordinate $\theta \in [0, 2\pi)$ and time point $t > 0$.

Similar to Eqs (3.15) and (3.16), we use the kernel derivatives to compute the different surface curvatures of the amoeba tube. For a smooth contour in a 2D space, the tangent direction at each position on the contour is well defined and, therefore, also the contour curvature. In the case of surfaces in a 3D space, this becomes more complicated, since the curvature varies with respect to the sectional curvature along a tangent vector. For this reason, there are multiple surface curvatures defined in the 3D case, e.g., the Gaussian curvature or the mean curvature. First, we introduce the following coefficients E, F, G and L, M, N for a given position on the amoeba tube surface with $\theta \in [0, 2\pi)$ and $t > 0$:

$$\begin{aligned} E(\theta,t) &= \frac{\partial}{\partial\theta} f(\theta,t) \cdot \frac{\partial}{\partial\theta} f(\theta,t), \qquad & L(\theta,t) &= \nu(\theta,t) \cdot \frac{\partial^2}{\partial\theta^2} f(\theta,t), \\ F(\theta,t) &= \frac{\partial}{\partial\theta} f(\theta,t) \cdot \frac{\partial}{\partial t} f(\theta,t), & M(\theta,t) &= \nu(\theta,t) \cdot \frac{\partial^2}{\partial\theta\partial t} f(\theta,t), \\ G(\theta,t) &= \frac{\partial}{\partial t} f(\theta,t) \cdot \frac{\partial}{\partial t} f(\theta,t), & N(\theta,t) &= \nu(\theta,t) \cdot \frac{\partial^2}{\partial t^2} f(\theta,t). \end{aligned}$$

where ν describes a unit normal vector defined by

$$\nu(\theta,t) = \frac{\frac{\partial}{\partial\theta}f(\theta,t) \times \frac{\partial}{\partial t}f\theta,t)}{\|\frac{\partial}{\partial\theta}f(\theta,t) \times \frac{\partial}{\partial t}f(\theta,t)\|_2}.$$

In differential geometry, E, F, G and L, M, N are known as coefficients of the first and second fundamental forms, respectively. Then, we compute the Gaussian curvature K and the mean curvature H by the following formulas:

$$K = \frac{LN - M^2}{EG - F^2},$$
$$H = \frac{LG - 2MF + NE}{2(EG - F^2)}.$$

For every point on the amoeba tube surface, we obtain the maximum curvature k_1 and minimum curvature k_2 (so-called principal curvatures) by

$$\kappa_{1,2} = H \pm \sqrt{H^2 - K}.$$

While the Gaussian curvature is given by the product of the principle curvatures, i.e., $K = \kappa_1\kappa_2$, the mean curvature is given by $H = (\kappa_1 + \kappa_2)/2$. In case of a right cylinder with radius $r > 0$, the principal curvatures are given by $\kappa_1 = 1/r$ and $\kappa_2 = 0$. For this reason, the Gaussian curvature is equal to zero at every position on the cylinder, whereas the mean curvature at each position is equal to $1/(2r)$.

In Fig 3.4, we used the above approach to compute the different surface curvatures for the artificial case of a pulsating circle: the Gaussian curvature (panel (A)), the mean curvature (panel (B)), and the principal curvatures (panels (C) and (D)). Since the contour dynamics consists of circles at every time, the curvature in horizontal direction is always positive. For this reason, the first principal curvature (maximum curvature) takes only positive values (panel C). At the thin part of the amoeba tube ($t = 0s$ and $t = 50s$), the maximum curvature is defined in horizontal direction, while it is defined in vertical direction for the thick part of the amoeba tube ($t = 25s$ and $t = 75s$). Vice versa, the minimum curvature (panel (D)) is defined vertically at $t = 0s$ and $t = 50s$, taking negative values due to the concavity, whereas for $t = 25s$ and $t = 75s$ the minimum curvature is defined horizontally. Finally, for these kind of contour dynamics, we observe a qualitative similarity between the Gaussian curvature (panel (A)) and the mean curvature (panel (B)). However, both curvatures are quantitatively different, since the Gaussian curvature is defined by the product of the principal curvatures.

In summary, we applied the GPR to the entire contour dynamics instead of a single contour. This way, we obtained a smooth 3D (amoeba tube) representation. Analogous to the 2D case, we used derivatives of the kernel functions to obtain estimates of the different surface curvatures. However, from the example in Fig 3.4, we see that a membrane expansion, succeeded by a fast retraction at the same position, creates a higher surface curvature. Hence, we would mainly identify expansion

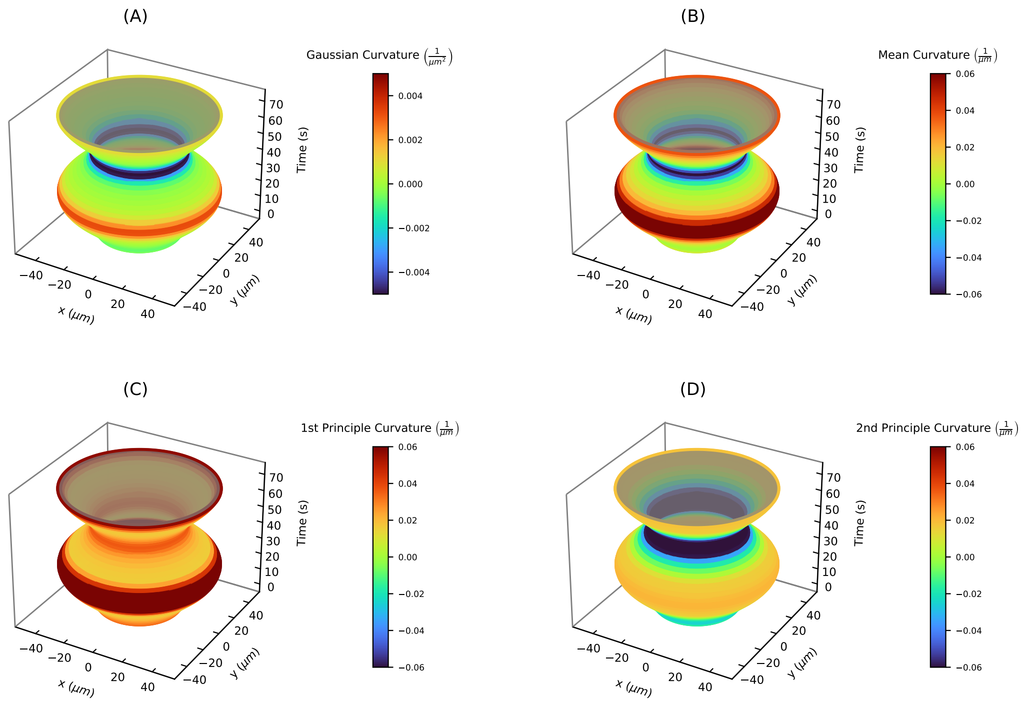


Figure 3.4. Surface curvatures of artificial contour dynamics of a pulsating circle: (A) the Gaussian curvature, **(B)** the mean curvature, **(C)** the 1st principal curvature (maximum curvature), **(D)** and the 2nd principal curvatures (minimum curvature).

events with no lasting effects on the overall contour dynamics. In the artificial case of a circle moving with constant velocity in one direction, the Gaussian curvature is equal to zero at every position of the skewed cylindrical amoeba tube surface. By changing the underlying velocity of the circle, the Gaussian curvature is not affected. This example shows that the above approach cannot be adequately used to identify or classify membrane expansions. Another disadvantage of this approach are high computational costs due to high-dimensional covariance matrices. For an exemplary data set of 500 contours, recorded with a frame rate of $\delta t = 1s$, and 400 segmentation points per contour, we would obtain a covariance matrix Σ with a dimension of 200.000×200.000. Hence, the above method becomes only feasible on a standard computer if a smaller number of contours is chosen, e.g., $n \approx 30$. For this reason, one might consider using a (temporal) kernel function with compact support, e.g., the Wendland kernel function, to obtain sparse matrices.

Later on in Chapter 4, we present a more effective and fully automated approach to identify protrusion and retraction events. This approach applies the 2D case of the GPR presented in Section 3.1.2 to obtain smooth cell contours. Then, consecutive contours are mapped onto each other with respect to so-called regularized contour flows instead of using the spatio-temporal GPR framework described before.

3.2 Geometric contour flows

In this section, we provide the mathematical background for different geometric contour flows which are used in our amoeboid cell motility model in Chapter 5: the curve-shortening flow (CSF), the gradient flow of the area functional, and the area-preserving curve-shortening flow (APCSF). The sections 3.2.1 and 3.2.2 are based on [93, 94]. For more details regarding the APCSF, see [95, 96, 97].

3.2.1 Curve-shortening flow.

Similar to the parametrization of the amoeba tube representation in Eq (3.18), the flow of a contour is written as

$$\begin{aligned} \Phi : [0,T] \times [0,2\pi) &\to \mathbb{R}^2, \\ (t,\tau) &\mapsto \Phi(t,\tau). \end{aligned} \tag{3.21}$$

Then, the arc length of the contour at time t is given by the following functional

$$L(t) = L\left(\Phi(t,\cdot)\right) = \int_0^{2\pi} \left\| \frac{\partial \Phi(t,\tau)}{\partial \tau} \right\|_2 \, d\tau. \tag{3.22}$$

In the following, we will write τ in the context of general curve parametrizations (see Eq (3.21)). In contrast, we will write s and θ when the curve is parametrized with respect to the arc length and the normalized arc length, respectively.

We are now interested in that flow which decreases the arc length of a contour in the direction of steepest descent, the so-called gradient flow of the arc length functional. First, we deduce the following equality

$$\begin{aligned} \frac{\partial}{\partial t} L(t) &= \int_0^{2\pi} \frac{\frac{\partial \Phi(t,\tau)}{\partial \tau}}{\left\| \frac{\partial \Phi(t,\tau)}{\partial \tau} \right\|_2} \cdot \frac{\partial^2 \Phi(t,\tau)}{\partial \tau \partial t} \, d\tau \\ &= \left[\frac{\frac{\partial \Phi}{\partial \tau}}{\left\| \frac{\partial \Phi}{\partial \tau} \right\|_2} \cdot \frac{\partial \Phi}{\partial t} \right]_0^{2\pi} - \int_0^{2\pi} \frac{\partial \Phi}{\partial t} \cdot \frac{\partial}{\partial \tau} \left(\frac{\frac{\partial \Phi}{\partial \tau}}{\left\| \frac{\partial \Phi}{\partial \tau} \right\|_2} \right) d\tau \\ &= 0 - \int_0^{2\pi} \frac{\partial \Phi(t,\tau}{\partial t} \cdot \frac{\partial \vec{t}(t,\tau)}{\partial \tau} \, d\tau \end{aligned}$$

where $\vec{t}(t,\tau) \in \mathbb{R}^2$ denotes the unit tangent vector with respect to the curve parameter $\tau \in [0,2\pi)$ and time t. Similarly, we denote the unit normal vector by $\vec{n}(t,\tau) \in \mathbb{R}^2$ and the curvature by $\kappa(t,\tau) \in \mathbb{R}$. By reparametrizing the above integral with respect to the arc length $ds = \|\partial\Phi/\partial\tau\|_2 \, d\tau$, we obtain

$$\begin{aligned} \frac{\partial}{\partial t} L(t) &= - \int_0^{L(t)} \frac{\partial \Phi(t,s)}{\partial t} \cdot \frac{\frac{\partial \vec{t}(t,s)}{\partial \tau}}{\left\| \frac{\partial \Phi(t,s)}{\partial \tau} \right\|_2} \, ds \\ &= - \int_0^{L(t)} \kappa(t,s) \, \frac{\partial \Phi(t,s)}{\partial t} \cdot \vec{n}(t,s) \, ds \end{aligned}$$

Hence, the gradient flow of $L(t)$ is given by

$$\left\langle \frac{\partial \Phi(t,s)}{\partial t}, \vec{n}(t,s) \right\rangle = \kappa(t,s)$$

which is known as the curve-shortening flow (CSF). Oftentimes, the CSF is introduced by the following equation:

$$\frac{\partial \Phi(t,\tau)}{\partial t} = \kappa(t,\tau)\ \vec{n}_{\text{inw}}(t,\tau),$$

where $\vec{n}_{\text{inw}}(t,\tau) \in \mathbb{R}^2$ denotes the inward-pointing unit normal vector [93, 94, 95]. However, this formula is only well-defined for a curve without self-intersections, and if the time derivative $\partial\Phi/\partial t$ points toward the normal direction, which is usually not the case for general curve parametrizations.

From above, we see that the rate, with which the total arc length decreases under the CSF, is given by

$$\begin{aligned} \frac{\partial}{\partial t} L(t) &= -\int_0^{L(t)} \kappa^2\ \|\vec{n}_{\text{inw}}\|_2^2\ ds \\ &= -\int_0^{L(t)} \kappa^2\ ds \leq 0, \end{aligned}$$

where $\int_0^{L(t)} \kappa^2\ ds$ is called the total squared curvature. Based on this rate, we see that the arc length of a contour decreases monotonically under the CSF until the contour evolves to a single point. In the case of a shrinking circle, the curvature grows indefinitely since the curvature is inversely proportional to the radius. For this reason, the CSF becomes faster over time. Furthermore, a simple closed curve, i.e., a closed curve without self-intersections, remains simple under the CSF. The same holds true for a smooth curve [94].

3.2.2 Gradient flow of the area functional.

Similarly to the above computations, we now compute the gradient flow of the area functional defined by

$$\begin{aligned} A(t) = A(\Phi(t,\cdot)) &:= \int_0^{2\pi} \Phi^{(x)} \frac{\partial \Phi^{(y)}}{\partial \tau}\ d\tau && (3.23) \\ &= \int_0^{2\pi} \Phi^{(y)} \frac{\partial \Phi^{(x)}}{\partial \tau}\ d\tau, && (3.24) \end{aligned}$$

where $\Phi^{(x)}(t,\tau) \in \mathbb{R}$ and $\Phi^{(y)}(t,\tau) \in \mathbb{R}$ denote the x- and y-component of $\Phi(t,\tau) \in \mathbb{R}^2$, respectively. Then, by following the steps described in [94], the first variation of $A(\Phi(t,\cdot))$ is given by

$$\delta A(\Phi(t,\cdot), h) = \frac{d}{d\varepsilon} A(\Phi(t,\cdot) + \varepsilon h)\bigg|_{\varepsilon=0}$$

$$
\begin{aligned}
&= \frac{d}{d\varepsilon}\left(\int_0^{2\pi}\left(\Phi^{(x)}+\varepsilon h^{(x)}\right)\frac{\partial\left(\Phi^{(y)}+\varepsilon h^{(y)}\right)}{\partial\tau}\,d\tau\right)\Bigg|_{\varepsilon=0}\\
&= \frac{d}{d\varepsilon}\left(\int_0^{2\pi}\left(\Phi^{(x)}+\varepsilon h^{(x)}\right)\left(\frac{\partial\Phi^{(y)}}{\partial\tau}+\varepsilon\frac{\partial h^{(y)}}{\partial\tau}\right)\,d\tau\right)\Bigg|_{\varepsilon=0}\\
&= \frac{d}{d\varepsilon}\Bigg(\int_0^{2\pi}\Phi^{(x)}\frac{\partial\Phi^{(y)}}{\partial\tau}+\varepsilon\Phi^{(x)}\frac{\partial h^{(y)}}{\partial\tau}+\varepsilon h^{(x)}\frac{\partial\Phi^{(y)}}{\partial\tau}\\
&\qquad\qquad +\varepsilon^2 h^{(x)}\frac{\partial h^{(y)}}{\partial\tau}\,d\tau\Bigg)\Bigg|_{\varepsilon=0}\\
&= \int_0^{2\pi}\Phi^{(x)}\frac{\partial h^{(y)}}{\partial\tau}+h^{(x)}\frac{\partial\Phi^{(y)}}{\partial\tau}\,d\tau\\
&= \left[\Phi^{(x)}h^{(y)}\right]_0^{2\pi}-\int_0^{2\pi}h^{(y)}\frac{\partial\Phi^{(x)}}{\partial\tau}\,d\tau+\int_0^{2\pi}h^{(x)}\frac{\partial\Phi^{(y)}}{\partial\tau}\,d\tau\\
&= \int_0^{2\pi}h^{(x)}\frac{\partial\Phi^{(y)}}{\partial\tau}-h^{(y)}\frac{\partial\Phi^{(x)}}{\partial\tau}\,d\tau\\
&= \int_0^{2\pi}\left\langle h,\begin{pmatrix}\frac{\partial\Phi^{(y)}}{\partial\tau}\\ -\frac{\partial\Phi^{(x)}}{\partial\tau}\end{pmatrix}\right\rangle\,d\tau=\int_0^{2\pi}\left\langle h,\begin{pmatrix}\frac{\partial\Phi^{(x)}}{\partial\tau}\\ \frac{\partial\Phi^{(y)}}{\partial\tau}\end{pmatrix}^{\perp}\right\rangle\,d\tau\\
&= \int_0^{L(t)}\left\langle h,\frac{\left(\frac{\partial\Phi}{\partial\tau}\right)^{\perp}}{\|\frac{\partial\Phi}{\partial\tau}\|}\right\rangle\,ds=\int_0^{L(t)}\left\langle h,\vec{t}^{\perp}\right\rangle\,ds\\
&= \int_0^{L(t)}\langle h,\vec{n}\rangle\,ds
\end{aligned}
$$

and, therefore,

$$
\delta A(\Phi,h)=-\int_0^{L(t)}\langle h,\vec{n}_{\text{inw}}\rangle\,ds. \tag{3.25}
$$

Thus, by inserting the $h=\partial\Phi/\partial t$, we see that

$$
\frac{\partial\Phi}{\partial t}=\vec{n}_{\text{inw}} \tag{3.26}
$$

is a gradient flow of the area functional $A(\Phi)$ [94], where the corresponding rate is given by

$$
\frac{\partial A(\Phi)}{\partial t}=-\int_0^{L(t)}\|\vec{n}_{\text{inw}}\|_2^2\,ds=-L(t).
$$

Of note, Eq (3.26) is only valid if the time derivative $\partial\Phi/\partial t$ points toward the normal direction.

Similarly, we can compute the gradient flow of the following "area adjustment functional":

$$
F(\Phi):=(A(\Phi)-A_{\text{ref}})^2\rightarrow\min,
$$

where $A_{\text{ref}} \in \mathbb{R}^+$ denotes the desired reference area. By using Eq (3.25), we compute the first variation

$$\delta F(\Phi, h) = -2\left(A(\Phi) - A_{\text{ref}}\right) \int_0^{L(t)} \langle h, \vec{n}_{\text{inw}} \rangle \, ds.$$

The gradient flow of the above area adjustment functional is then given by

$$h = \frac{\partial \Phi}{\partial t} = -2\left(A(\Phi) - A_{\text{ref}}\right) \vec{n}_{\text{inw}}, \tag{3.27}$$

where the corresponding rate is

$$\frac{\partial F}{\partial t} = -2(A(\Phi) - A_{\text{ref}}) \, L(t).$$

Noteworthy, this gradient flow affects the contour in normal direction only. While the CSF preserves (and even enforces) the regularity of a closed curve, the gradient flows from Eqs (3.26) and (3.27) can produce self-intersections [94]. Later on, in Section 5.2.2, we will introduce a further geometric flow called area adjustment flow (AAF). In contrast to the gradient flow from Eq (3.27), the AAF is shape-preserving to prevent the occurrence of self-intersections.

3.2.3 Area-preserving curve-shortening flow.

Now, we introduce the gradient flow of the arc length functional from Eq (3.22) under an additional area-preserving constraint $A(t) := A(\Phi(t,\cdot)) = A(0)$ for all $t > 0$, with contour area $A(t)$ defined as in Eq (3.23). This gradient flow is called the area-preserving curve-shortening flow (APCSF) and is given by

$$\left\langle \frac{\partial \Phi(t,\tau)}{\partial t}, \vec{n}_{\text{inw}}(t,\tau) \right\rangle = \kappa(t,\tau) - \frac{2\pi}{L(t)}. \tag{3.28}$$

Oftentimes, the following equation is used to define the APCSF [95, 96, 97]:

$$\frac{\partial \Phi(t,\tau)}{\partial t} = \left(\kappa(t,\tau) - \frac{2\pi}{L(t)}\right) \vec{n}_{\text{inw}}(t,\tau).$$

Again, $\vec{n}_{\text{inw}}$ is only well-defined when the underlying closed curve is simple, i.e., without self-intersections.

The APCSF evolves every contour to a circle of the same area, minimizing the contour arc length to $L(t) \xrightarrow{t\to\infty} 2\sqrt{\pi A(0)}$ while maintaining its area. By inserting the definition of the APCSF into Eq (3.25), if follows that the APCSF is indeed area-preserving:

$$\begin{aligned} A'(t) &= -\int_0^{L(t)} \langle h(t,s), \vec{n}_{\text{inw}}(t,s) \rangle \; ds \\ &= -\int_0^{L(t)} \kappa(t,s) - \frac{2\pi}{L(t)} \; ds \end{aligned}$$

$$\begin{aligned}
&= -\int_0^{L(t)} \kappa(t,s)\,ds + \int_0^{L(t)} \frac{2\pi}{L(t)}\,ds \\
&= -2\pi + L(t)\frac{2\pi}{L(t)} = 0.
\end{aligned}$$

Now, we show that the total arc length is monotonically decreasing under the APCSF. First, we compute the decay of $L(t)$ under the APCSF via:

$$\begin{aligned}
L'(t) &= -\int_0^{L(t)} \kappa(t,s) \left\langle \frac{\partial \Phi(t,s)}{\partial t}, \vec{n}_{\text{inw}}(t,s) \right\rangle ds \\
&= -\int_0^{L(t)} \kappa^2(t,s) - \kappa(t,s)\frac{2\pi}{L(t)}\,ds \\
&= -\int_0^{L(t)} \kappa^2(t,s)\,ds + \frac{2\pi}{L(t)}\int_0^{L(t)} \kappa(t,s)\,ds \\
&= -\int_0^{L(t)} \kappa^2(t,s)\,ds + \frac{4\pi^2}{L(t)} =: \bigstar.
\end{aligned}$$

By using the following equality

$$\begin{aligned}
-\int_0^{L(t)} \left(\kappa(t,s) - \frac{2\pi}{L(t)}\right)^2 ds &= -\int_0^{L(t)} \kappa^2(t,s) - \frac{4\pi}{L(t)}\kappa(t,s) + \frac{4\pi^2}{L(t)^2}\,ds \\
&= -\int_0^{L(t)} \kappa^2(t,s)\,ds + \frac{8\pi^2}{L(t)} - \frac{4\pi^2}{L(t)} \\
&= -\int_0^{L(t)} \kappa^2(t,s)\,ds + \frac{4\pi^2}{L(t)} = \bigstar,
\end{aligned}$$

we have show that the arc length is monotonically decreasing under the APCSF:

$$L'(t) = -\int_0^{L(t)} \left(\kappa(t,s) - \frac{2\pi}{L(t)}\right)^2 ds \le 0,$$

with $L'(t) = 0$ if and only if the current contour is a circle with radius $r = \frac{1}{\kappa} = \frac{L}{2\pi}$.

3.3 Stochastic processes

In this section, we introduce two stochastic processes, the Hawkes process (HP) and the Ornstein-Uhlenbeck process (OUP), which are both used as underlying driving force in our amoeboid motility model presented later on in Chapter 5. See [98, 99, 100], for more details about the HP. Regarding the OUP and related diffusion processes, we recommend [101].

3.3.1 Hawkes process

The Hawkes process is named after Alan G. Hawkes and describes a self-exciting Poisson point process [102, 103, 104]. In contrast to a standard (i.e. not self-exciting) Poisson point process, the underlying intensity of a HP is temporarily increased after incoming events. Just like the Poisson point process, the HP can be interpreted as counting process $(N_t, t \geq 0)$, i.e., a non-decreasing stochastic process taking non-negative integer values.

Definition 3.3.1 (Hawkes Process). *A Hawkes process is defined as a counting process $(N_t, t \geq 0)$ with conditional intensity*

$$\begin{aligned} \lambda(t) &= \mu(t) + \int_0^t g(t-s)\, dN(s) \\ &= \mu(t) + \sum_{i:t_i<t} g(t-t_i), \end{aligned} \tag{3.29}$$

with event times $\{t_1, t_2, \dots\}$, background intensity function $\mu : \mathbb{R}^+ \to \mathbb{R}^+$, and kernel function $g : \mathbb{R}^+ \to \mathbb{R}^+$, characterizing the positive influence of past events (ancestors) on the emergence of future events (descendants).

The expected number of offspring for the same parent event is given by

$$m = \int_0^\infty g(t)dt.$$

Crucially, the kernel function g must be chosen such that the expected number of offspring is $0 < m < 1$. Otherwise, if every event is expected to have more than one offspring ($m > 1$), the total number of HP events becomes infinite. By choosing, for example, the commonly-used exponential kernel

$$g(t) := \alpha e^{(\beta t)}, \quad \alpha, \beta > 0,$$

we obtain $m = \alpha/\beta$. Thus, for this kernel, one should choose α and β such that $\alpha < \beta$. For $g(t) = 0$, we obtain a standard Poisson point process with background intensity $\lambda(t) = \mu(t)$ and no offspring ($m = 0$).

The HP intensity from Eq (3.29) is a function of time only. For our amoeboid motility model in Chapter 5, we will use a spatio-temporal version of the HP, see [100] for more details.

3.3.2 Ornstein-Uhlenbeck process

The Ornstein-Uhlenbeck process (OUP) is a stochastic process, which was introduced in 1930 by Leonard Ornstein and George E. Uhlenbeck [105]. It consists of two terms: a deterministic mean reversion term and a stochastic diffusion term driven by a Brownian motion W_t. While the OUP was initially used to describe the diffusion of particles, it became increasingly important in other fields such as financial mathematics and biophysics. The OUP is defined as follows.

Definition 3.3.2 (One-dimensional Ornstein-Uhlenbeck process)**.** *The stochastic process $(X_t, t \geq 0)$ defined by the following stochastic differential equation (SDE) is called Ornstein-Uhlenbeck process (OUP):*

$$dX_t = -aX_t\, dt + b\, dW_t, \quad X(0) = x_0,$$

with initial state $x_0 \in \mathbb{R}$, mean reversion rate $a > 0$, diffusion rate $b > 0$, and W_t denoting the one-dimensional Brownian motion.

The name of "mean reversion rate" of a arises from a more general definition of the OUP based on the SDE:

$$dX_t = a(\mu - X_t)\,dt + b\,dW_t, \quad X(0) = x_0.$$

For this case, the OUP has the following expectation and covariance:

$$\begin{aligned}\mathbb{E}[X_t] &= x_0 e^{-at} + \mu\left(1 - e^{-at}\right), \\ \mathrm{Cov}[X_t, X_s] &= \frac{b^2}{2a}\left(e^{-a|t-s|} - e^{-a(t+s)}\right).\end{aligned}$$

To obtain a variance term depending on the diffusion rate b only, i.e., $\mathbb{V}[X_t] = b^2$, the OUP is sometimes reparametrized by the following SDE:

$$dX_t = -aX_t\,dt + b\sqrt{2a}\,dW_t, \quad X(0) = 0. \tag{3.30}$$

Later on in Chapter 5, we will use a multivariate version of Eq (3.30) to define the protrusion-generating term in our amoeboid motility model. In contrast to Eq (3.30), this version will affect the entire contour instead of a single particle. Furthermore, it will be based on a correlated and periodic noise on the contour instead of a Brownian motion.

Analyzing amoeboid cell motility 4

To advance the mathematical description of amoeboid motility, we envision that current efforts of mechanism-based modeling are complemented by a more systematic, data-driven approach. This requires a mathematical framework that allows us to systematically develop a quantitative model based on experimental data. Such a framework should rely on observables that encode the key characteristics of amoeboid motility and, at the same time, are readily accessible experimentally. Trajectories of the center-of-mass of the cell can be easily recorded in large amounts from low-resolution bright-field microscopy data but reflect only very limited, integral information on the entire process. The intracellular signaling pathways and the cytoskeletal mechanisms, on the other hand, are difficult to access and knowledge on this part of the system remains highly incomplete. We therefore concentrate on the cell shape as the central reference quantity. The cell shape is fully accessible by standard microscopy techniques and can be easily recorded with sufficient spatial and temporal resolution. Moreover, its dynamic evolution implicitly reflects the intracellular processes and determines the center-of-mass trajectory of the cell.

In our long-term quest for a quantitative, data-driven model of amoeboid motility, several steps are required: First, the development of a mathematical framework for the description of experimentally observed shape dynamics; second, the design of a model of the contour dynamics that predicts realistic shape evolutions; and finally, the incorporation of mechanistic information on key intracellular processes as the driving determinants of the contour dynamics. These become accessible by imaging of fluorescently tagged fusion proteins and by more advanced methods, such as knock-sideways and optogenetic approaches. Here, existing mechanistic models may provide a useful basis and might merge into a joint modeling concept.

In this chapter, we concentrate on the first aspect of the above agenda. We provide a mathematically well-defined approach that allows for a detailed analysis of the complex, multifaceted contour dynamics of amoeboid cells. A key ingredient is the concept of regularized flows between contours that define an evolution of virtual markers in time. Using contour flows, we define a coordinate system on the evolving contours (strongly regularized case) and approximate local quantities of interest (weakly regularized case). While the strongly regularized flow is used to define trajectories of virtual markers over the entire time span, the weakly regularized flow is used to obtain information on local membrane changes, which is mapped subsequently onto the global flow representing the coordinate system. This separation between global and local flows is an essential feature of our approach.

Established approaches to describe virtual markers on an evolving contour most commonly include level set methods (LSM), e.g., in [27], where the cell contour dynamics was additionally decomposed into a translation of the entire cell and the deformation of its contours. In [29], the LSM was compared to a mechanistic spring model penalizing a dense concentration of virtual markers. While the mechanistic model provides better computation times than the LSM, it shows limitations regarding topological mapping violations during strong shape deformations. Alternatively, electrostatic field equations were successfully employed, tackling mapping violation issues while providing better computation times [28]. However, it does not provide trajectories of virtual markers over the entire time interval in its current form. In [19], a mapping was chosen which minimizes the sum of squared distances between virtual markers (so-called mean squared displacement), while enforcing an evenly-spaced distribution of virtual markers. Our work aims at (i) presenting a theoretical framework for the analysis of marker dynamics and (ii) combining the respective merits of the above approaches into a single method.

In the spirit of previous approaches, we rely on the widely used concept of kymographs to graphically represent the space-time dynamics of different geometric quantities, such as the speed of membrane displacement or the local curvature, along the cell contour. In this context, we propose a novel criterion based on a single dynamic quantity for defining membrane expansions/contractions. Prior approaches to identify membrane protrusions relied on the simultaneous matching of multiple criteria, such as critical values of curvature, protrusion speed, and pseudopod lifetime [23] or identified protrusion events as points in time only [19], without providing major protrusion properties, such as area growth rate, shape, and others. In [25], pseudopods were detected by using a hierarchical clustering algorithm in which individual membrane extensions are connected based on direction and continuity in space and time. Furthermore, image skeletonization on contours was used to identify and characterize pseudopods in an automated way [26]. Unfortunately, all existing approaches have in common an undesirable blending of the protrusion-defining criteria with their numerical implementation. This makes it difficult to discern biological effects from numerical artifacts. We instead first define our criterion in mathematical terms and only subsequently implement it numerically, allowing us to control numerical errors and avoid artifacts.

The chapter is organized as follows. We first develop the mathematical framework and introduce the concept of regularized contour flows. We then illustrate our approach in applications to experimental recordings of the social amoeba *Dictyostelium discoideum*, a widely used model organism for the study of amoeboid motility. Finally, based on the results of our analysis, we illustrate in the Discussion section how we envision the next steps towards a quantitative, data-driven model of amoeboid motility, based on a point process of the protrusive activity.

4.1 Data acquisition and image segmentation

Experiments were performed with AX2 cells of the social amoeba *Dictyostelium discoideum*. As a marker for filamentous actin, cells expressed fluorescently tagged Lifeact (C-terminally fused with mRFP, plasmid kindly provided by Igor Weber, Zagreb, HR). They were grown at $20°$C in liquid culture flasks containing HL5 medium including glucose (Formedium, Hunstanton, GB) and 10 μg/ml G-418 sulfate (Cayman Chemical Company, US) as a selection marker. Before each experiment, cells were harvested from culture flasks and grown in 25 ml overnight shaking cultures at 180 rpm under otherwise identical conditions. Afterward, nutrients were removed by centrifugation and washing of the cell pellet with Sørensen phosphate buffer at pH 6 (14.7 mM KH_2PO_4, Merck, Darmstadt, DE; 2 mM $Na_2HPO_4 \times H_2O$, Merck, Darmstadt, DE). Then, cells were resuspended in fresh buffer and droplets were formed in a Petri dish to initiate the streaming processes.

For image acquisition cells were transferred after 5 hours to a glass bottom dish (Fluorodish, ibidi GmbH, Martinsried, DE). During imaging, they were kept in Sørensen phosphate buffer at $20°$C. Images were taken with a Zeiss LSM780 laser scanning confocal microscope (Carl Zeiss AG, Oberkochen, DE) at a frame rate of one image per second, using a $63\times$ or $40\times$ oil immersion objective. Fluorophores were excited at 651 nm and emission was recorded between 562 nm and 704 nm. For details see also [14], where the same data set was used in a different context.

Fluorescence image (8-bit gray scale) were segmented using a modified version of the active contour (snake) algorithm described in [24, 19]. Based on this algorithm, we parameterized the cell boundary in each frame by a closed string of $M = 400$ equidistant nodes. As frames were taken at discrete time points $t_0, \ldots, t_{K-1}$ with equal time difference $\delta t = t_{k+1} - t_k = 1$ sec, we denoted the discrete representation of the cell contour in a given frame at time t_k as γ_k with supporting points

$$(x_{k,0}, y_{k,0}), \ldots, (x_{k,M-1}, y_{k,M-1}) \in \mathbb{R}^2. \tag{4.1}$$

In this chapter, the data sets consists of $K = 500$ to 1000 time frames. For later reference, we set $t_0 = 0$ and $t_{K-1} = T$.

4.2 Mathematical framework: Regularized contour flows

4.2.1 Obtaining smooth contour estimates via GPR

We used a real-valued smoothing spline for the x and y coordinates based on the GPR using a Poisson kernel, for details see Section 3.1. This yielded a parametrization of the contour Γ_k at time frame $t_k = k \cdot \delta t$

$$\Phi_k : [0, 2\pi) \ni \theta \mapsto (x(\theta), y(\theta)) \in \mathbb{R}^2 \tag{4.2}$$

in terms of a finite sum of smooth kernels (see e.g. [82, 83] for details)

$$x(\theta) = \sum_{m=0}^{M-1} c_m P(\theta_m - \theta), \qquad y(\theta) = \sum_{m=0}^{M-1} d_m P(\theta_m - \theta)$$

where P is a suitably scaled Poisson kernel. Support points were chosen to correspond to normalized secant length along the contour

$$\theta_{k,m} = \frac{2\pi \sum_{i=0}^{m} d_{k,i}}{\sum_{i=0}^{M-1} d_{k,i}}, \quad d_{k,0} = 0, \quad d_{k,i} = \left((x_{k,i} - x_{k,i-1})^2 + (y_{k,i} - y_{k,i-1})^2\right)^{1/2}$$

for $m = 0, \dots, M-1$ and $k = 0, \dots, K-1$, see Section 3.1.2 for details. We parametrized the contour in the mathematical positive sense, i.e., the interior of the cell is on the left when going around the contour with increasing θ. In the numerical implementation, we used the rescaled arc length coordinates, which we denoted again by θ. This gives

$$\|\partial_\theta \Phi_k(\theta)\| \equiv \frac{L_k}{2\pi}, \quad \text{with} \quad L_k = \int_0^{2\pi} \|\partial_\theta \Phi_k(\theta)\| \mathrm{d}\theta, \tag{4.3}$$

denoting the length of contour Γ_k. Note that Φ_k is only uniquely determined up to a phase shift, i.e., for every Φ_k and τ, also $\Phi_{k,\tau}(\cdot) = \Phi_k(\cdot + \tau)$ is a valid parametrization of Γ_k. The phase shift was chosen by an additional requirement in the next section.

The smooth parametrization Φ_k allowed us compute local quantities along the contour, e.g., its curvature

$$\kappa = \frac{R_{\pi/2} \partial_\theta \Phi_k(\theta) \cdot \partial_\theta^2 \Phi_k(\theta)}{\|\partial_\theta \Phi_k(\theta)\|^3}, \tag{4.4}$$

where $R_{\pi/2}$ is an anti-clockwise rotation by $\pi/2$. As global quantity, we determined the center of mass C_k of contour Φ_k as

$$C_k = \frac{1}{2\pi} \int_0^{2\pi} \Phi_k(\theta) d\theta. \tag{4.5}$$

Since the segmentation points were rather densely spaced over the contour, they well constrained the smooth contours. Based on the kernel representation, all geometric quantities, such as arc length and curvature were defined intrinsically for each contour, and may be easily computed numerically and with high precision.

Connecting the contours along the time axis, however, is not intrinsically well defined, and is bound to choices. In a first step, we constructed a global coordinate flow, which served as a reference frame for further local flows to be defined subsequently. In the limit of infinitely densely sampled contours, this global coordinate system results in a mapping

$$\Phi : [0, T] \times [0, 2\pi) \to \mathbb{R}^2, \quad (t, \theta) \mapsto \Phi(t, \theta).$$

Vice versa, any coordinate system defines a coordinate flow over the tube of contours, i.e., the cell contours in 2d mapped into a 3d space-time coordinate system. If $p_0 = (x_0, y_0)$ is a point on the first contour at $t = 0$ and if θ_0 is its arc length coordinate, then $t \mapsto \Phi(t, \theta_0)$ corresponds the movement of p_0 over the space–time tube of contours, see Fig 3.3. Of note, this coordinate flow is a theoretical

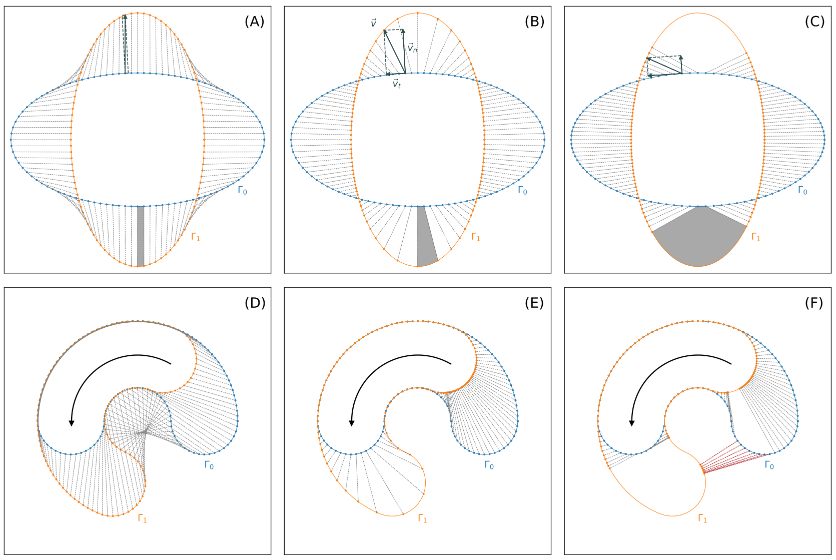


Figure 4.1. Virtual marker flows for two test cases w.r.t different degrees of regularization: strongly regularized **(A, D)**, weakly regularized **(B, E)**, and non-regularized **(C, E)**. In panel (A)–(C), a stretching effect is illustrated as gray area. Furthermore, mapping violations are highlighted as red lines **(F)**.

construct that allows us to analyze amoeboid contour dynamics and should not be misinterpreted as a flow of specific membrane lipids or proteins. Nevertheless, such a global coordinate system is useful and allows for graphical visualization of the contours and any locally defined quantity in form of a kymograph.

4.2.2 Biophysical motivation & description of our contour dynamics approach

After obtaining smooth contours at discrete time steps the question of how to link them in time remains open. Mapping a marker on one contour to its nearest point on the consecutive contour is one desirable feature; achieving a regularly spaced distribution of markers on the consecutive contour is another desirable feature. Typically, a regularization parameter is used to balance the two features. In Fig 4.1, mappings defined by differently regularized flows of virtual markers are shown for two consecutive contours: a strong regularization (left column), a weak regularization (middle column), and no regularization at all (right column). While the strongly regularized flow preserves the distances of neighboring virtual markers, the non-regularized flow is defined by the shortest paths from the first contour to the second.

Additionally, in panel (B), the velocity $\vec{v}(\theta)$ that propagates a virtual marker θ over a particular distance to the next contour, is illustrated as well as its decomposition into an outward-pointing normal velocity $\vec{v}_n(\theta)$ and a tangential velocity $\vec{v}_t(\theta)$.

For the weakly and non-regularized cases, a stretching effect of virtual markers can be observed for expanding parts (gray area), whereas clustering effects of virtual markers occur at contracting regions. Assuming infinitesimally small time steps, the "stretching rate" at arc length parameter s that arises from transitioning from one contour to the next one is given by

$$d(s) = \frac{\partial v_t(s)}{\partial s} + \kappa(s) \cdot v_n(s), \tag{4.6}$$

where v_t and v_n denote the (scalar) speed in tangential and normal direction, respectively, and κ the curvature. In the following sections, we show that this kind of "stretching rate", in the sequel termed instantaneous dilation rate, is a useful quantity to identify active regions of the contour during cell motility.

While the normal component of the velocity can be approximated easily by taking the normal distances from the first contour to the second divided by δt, the tangential component remains unknown and therefore requires additional attention. Naive flows, such as the shortest path flow from Fig 4.1 (right column) or the flow merely based on the normal component, can produce topological mapping violations. An example of such a mapping violation can be seen in panel (F) where the order of virtual markers on the second contour is inconsistent with the previous one.

In general, additional constraints on the tangent component are necessary with the aim of preserving an evenly-spaced distribution of virtual markers over time. In [29], these additional constraints were formulated as a mechanistic spring model, providing better computation times than the more commonly used LSM. However, it is mentioned that for strong shape deformations this mechanistic model easily fails because of mapping violations. This approach, as well as ours, is based on a cost functional minimizing the trajectories of virtual markers to the next contour while penalizing the distances of neighboring virtual markers. While in the mechanistic spring model the penalizing distance is measured in $\mathbb{R}^2$, the space of contours, our approach measures the penalizing distance in $[0, 2\pi)$, the space to parametrize the contour. This key difference resolves the issue of mapping violations during large deformations in our approach.

In A.2, contour mappings for each of these two regularization schemes are displayed showing the advantageous stability of our developed $\mathcal{S}^1$ regularization during membrane spikes and other large shape deformations.

In A.3, a selection of contour mappings between two frames with larger shape deformations are displayed. The underlying microscopy data was recorded with an imaging rate of $\delta t \approx 3.13s$.

Our method combines and improves different aspects of previous approaches:

- Spatio-temporal tracking of a fixed number of virtual markers,
- Distances between virtual markers flexibly adjustable in terms of a single regularization,
- Faster than computationally expensive approaches such as LSM,

- Capability to capture large shape deformations/avoidance of mapping violations,
- Simplicity regarding the interpretation & number of parameters.

Another key feature of our approach is the usage of two different flows, separating the underlying coordinate system, defined by one (strongly regularized) flow, and the dynamic quantities of interest, which are defined separately, e.g., by a weakly regularized flow.

4.2.3 The maximal correlation coordinate system (MCCS)

The starting point are the parameterized contours Φ_k in Eq (4.2) for $k = 0, \dots, K-1$. To make the influence of the sampling rate more prominent, we also used the notation

$$\Phi(k\delta t, \theta) = \Phi(t_k, \theta) = \Phi_k(\theta).$$

As stated above, the parametrization of Φ_k is only determined up to a phase shift by Eq (4.3). We finally chose the phase shift and therefore the parametrization of Γ_{k+1} by minimizing the distance to the previous contour Γ_k in a mean squared sense, i.e.,

$$\tau_{k+1} = \operatorname*{argmin}_{\tau} \int_0^{2\pi} \|\Phi_{k+1}(\theta - \tau) - \Phi_k(\theta)\|^2 \mathrm{d}\theta. \tag{4.7}$$

We may alternatively interpret Eq (4.7) as optimizing the cross-covariance between the two contours when interpreted as vector-valued functions

$$\tau_{k+1} = \operatorname*{argmin}_{\tau} \int_0^{2\pi} \Phi_{k+1}(\theta - \tau) \cdot \Phi_k(\theta)\, \mathrm{d}\theta.$$

In the sequel, we used $\widetilde{\Phi}_{k+1}(\cdot) = \Phi_{k+1}(\cdot + \tau_{k+1})$ and omitted the tilde for ease of notation. Effectively, choosing the phase shift amounts to fixing the 'zero point' $\Phi_{k+1}(0)$ on Γ_{k+1}.

This is the coordinate system that from now on is used to represent the contour geometry, i.e., each scalar quantity q defined on the contour Γ_k with $q = q(t_k, (x, y))$ and $(x, y) \in \Gamma_k$ as function $(k, \theta) \mapsto q(k\delta t, \theta)$ of discrete time and continuous space can be represented w.r.t. the chosen coordinate system Φ.

4.2.4 The Eulerian and Lagrangian points of view

Any flow that maps the contour Γ_k into Γ_{k+1} is determined by a mapping that describes the translation along the contour ϕ_k. To ensure that $\theta \mapsto \phi_k(\theta)$ is a one-to-one map, we required in addition

$$\partial_\theta \phi_k(\theta) > 0. \tag{4.8}$$

An example of a mapping violation, i.e. a position θ with $\partial_\theta \phi_k(\theta) < 0$, is illustrated in Fig 4.1F, where the order of virtual markers on the following contour is inconsistent with the previous one. The iteration

$$\theta_{k+1} = \phi_k(\theta_k)$$

describes the trajectory $(\theta_k)_{k=0,\ldots,K-1}$ of the starting point at coordinate θ_0 on the first contour in our coordinate system. This approach to visualize the flow shall be called the Eulerian point of view, since it describes the translation vector field from Γ_k to Γ_{k+1} in the coordinate system of Γ_k:

$$\Phi_{k+1}(\phi_k(\theta)) = \Phi_k(\theta) + \delta t V_k(\theta). \tag{4.9}$$

The Lagrangian point of view instead describes the flow in the coordinate system it generates, which is different from our MCCS. Denote the coordinate of a point on the first contour by its angle coordinate θ_0, and let χ_k be the mapping of Γ_0 to Γ_k recursively defined by

$$\chi_{k+1}(\theta_0) = \phi_k(\chi_k(\theta_0)), \quad \chi_0(\theta_0) = \theta_0.$$

This gives $\chi_k(\theta_0) = \theta_k$ for $k = 0, \ldots, K-1$. The Lagrangian description $\Xi(t_k, \theta_0) \in \mathbb{R}^2$ is linked to the Eulerian description via

$$\Xi(t_k, \theta_0) = \Phi(t_k, \chi_k(\theta_0)).$$

Both points of view are useful to understand and describe a flow over the contour. The translation vector W_k in Lagrangian coordinates is simply

$$\Xi(t_{k+1}, \theta_0) = \Xi(t_k, \theta_0) + \delta t W_k(\theta_0)$$

and is linked to the Eulerian description via

$$V_k(\chi_k(\theta_0)) = W_k(\theta_0).$$

We used the functions ϕ_k, V_k and W_k interchangeably to specify the flow from Γ_k to Γ_{k+1}.

Transport along the flow. For any flow, we may define the instantaneous dilation rate LD of the flow. Considering two infinitesimally nearby points on contour Γ_k, we see that the local relative dilation/contraction factor is obtained from ϕ_k via

$$\mathrm{LD}_k(\theta)\delta t = \log(\partial_\theta \phi_k(\theta)). \tag{4.10}$$

Note that our global coordinate system MCCS induces a flow with uniform dilation rate. To describe the transport of a density under the flow, consider points on the contour Γ_k that are distributed according to a density $\mu_k(\theta)\mathrm{d}\theta$. Under the flow induced by ϕ_k this density changes according to

$$\mu_{k+1}(\phi_k(\theta))\mathrm{d}\theta = \frac{\mu_k(\theta)\mathrm{d}\theta}{\partial_\theta \phi_k(\theta)} \tag{4.11}$$

Starting from $\mu_0(\cdot) \equiv 1/(2\pi)$, this defines the transported density on all contours. In the Lagrangian picture this transport preserves the density $\mu_0(\cdot)$, which follows from the fact that by definition the starting angle does not change under the flow. The density μ_k can be written in Lagrangian coordinates as

$$\mu_k(\chi_k(\theta_0)) = \frac{1}{2\pi \cdot \partial_{\theta_0}\chi_k(\theta_0)}. \tag{4.12}$$

A regularizing family of flows. We next defined a family of *local* mappings ϕ_k between successive contours that yield a compromise between the shortest path flow and the uniform dilation coordinate flow, presented in the two end-member cases in Fig 4.1. Another suitable name for shortest path flow is reversed normal flow since the incoming trajectories under this flow are always orthogonal to the successive contour.

The mean squared velocity of the flow (with respect to a density μ_k) is given as

$$F_k[\phi_k] = \int_0^{2\pi} \left\| \frac{\Phi_{k+1}(\phi_k(\theta)) - \Phi_k(\theta)}{\delta t} \right\|^2 \mu_k(\theta)\mathrm{d}\theta = \int_0^{2\pi} \|V_k(\theta)\|^2 \mu_k(\theta)\mathrm{d}\theta. \tag{4.13}$$

The normal flow from contour Γ_k to Γ_{k+1} is the flow that departs from the first contour in the normal direction until it intersects with the second contour. The normal flow from Γ_{k+1} to Γ_k shall be called the reversed normal flow from Γ_k to Γ_{k+1}. This is the unconstrained minimizer of F_k. If there are no intersections of flow lines, it defines a one-to-one mapping between the two contours. In general, however, direct minimization of the functional F_k does not yield a valid flow because of self-intersections, and as a consequence, the induced map is multiple-valued. We, therefore, need to regularize the flow. A natural requirement is that the flow tends to enforce non-uniformly distributed points on contour Γ_k towards more uniformly distributed points on contour Γ_{k+1}. We proposed to quantify the degree of non-uniformity of a distribution $\mu(\theta)\mathrm{d}\theta$ by means of

$$U[\mu] = \int_0^{2\pi} \frac{\mathrm{d}\theta}{\mu(\theta)}. \tag{4.14}$$

Since any distribution satisfies $\int_0^{2\pi} \mu(\theta) = 1$ and $\mu(\theta) > 0$, the minimizer actually corresponds to the uniform distribution. Other measures of non-uniformity are possible, for instance the entropy

$$S[\mu] = \int_0^{2\pi} \mu(\theta)\log(\mu(\theta))d\theta.$$

This kind of regularization has been proposed in [106] by Otto, where the optimal flow is understood as a gradient flow of the entropy potential with respect to the Wasserstein transport distance of the marker density. Also very appealing, in this paper, we used the characterization in Eq (4.14), since it leads to a more readily tractable numeric quadratic minimization problem. In terms of the defining mapping ϕ_k, the functional U reads

$$U_k[\phi_k] = \int_0^{2\pi} \frac{\partial_\theta \phi_k(\theta)^2}{\mu_k(\theta)} \mathrm{d}\theta, \tag{4.15}$$

as follows from Eq (4.11). The regularized flow is defined as the flow that minimizes a compromise between both cost functions

$$\phi_{k,\lambda} = \underset{\phi_k}{\operatorname{argmin}} \; F_k[\phi_k] + \lambda U_k[\phi_k], \quad \lambda > 0. \tag{4.16}$$

Note that $H_k[\phi_k] = F_k[\phi_k] + \lambda U_k[\phi_k]$ depends on both, ϕ_k and the measure μ_k. When iterating over all contours, one needs to update the measure before optimizing the flow for the next time step. There are two end-member cases for the regularized flow:

- For large λ the optimal flow essentially immediately uniformizes the density of the initial contour. Thereafter, it is the uniform stretching flow that minimizes the mean square distances between the contours. This is precisely the coordinate flow defined before.
- For small λ the optimal flow allows for arbitrary local stretching rates to minimize the point-wise distance. Here the limit defines the regularized shortest path flow, respectively, the regularized reverse normal flow.

Note that straightforward pointwise minimization of the flow distance from Γ_k to Γ_{k+1} may lead to overlapping connections and hence singular mappings between the contours. If instead, we regularize with small λ, such overlaps are avoided.

The virtual marker picture.

For numerical implementation, we discretized the cost functionals using the concept of virtual markers on the contours. The virtual markers are a discretized version of the Lagrangian coordinates. Since μ_k is the transported density, the first cost functional for ϕ_k in the Lagrangian point of view using Eq (4.12) is given by

$$\begin{aligned} F_k[\phi_k] &= \frac{1}{2\pi} \int_0^{2\pi} \|V_k(\chi_k(\theta_0))\|^2 \mathrm{d}\theta_0 \\ &= \frac{1}{2\pi\delta t^2} \int_0^{2\pi} \|\Phi_{k+1}(\phi_k(\chi_k(\theta_0))) - \Phi_k(\chi_k(\theta_0))\|^2 \, \mathrm{d}\theta_0, \end{aligned} \tag{4.17}$$

while the second functional using Eq (4.12) is given by

$$U_k[\phi_k] = 2\pi \int_0^{2\pi} |\partial_{\theta_0} \chi_{k+1}(\theta_0)|^2 \mathrm{d}\theta_0. \tag{4.18}$$

See Section A.2 for details of the derivation. Both equations are well suited for a discrete numerical approximation for a given function f and initially equidistant $\theta_{0,i} = 2\pi i/N$ with $i = 0, \ldots, N-1$ using

$$\sum_{i=0}^{N-1} f(\theta_{0,i})(\theta_{0,i+1} - \theta_{0,i}) \simeq \int_0^{2\pi} f(\theta_0)\mathrm{d}\theta_0.$$

If we now approximate the continuous mapping ϕ_k by its discrete values on the virtual marker points $\theta_k = (\theta_{k,0}, \ldots, \theta_{k,N-1})$ with

$$\theta_{k+1,i} = \phi_k(\theta_{k,i}), \tag{4.19}$$

then the first cost function may be approximated as

$$F_k[\phi_k] \simeq F_k[\theta_{k+1}|\theta_k] = \frac{1}{N\delta t^2}\sum_{i=0}^{N-1} \|\Phi_{k+1}(\theta_{k+1,i}) - \Phi_k(\theta_{k,i})\|^2 \tag{4.20}$$

and the second cost function as

$$U_k[\phi_k] \simeq U_k[\theta_{k+1}|\theta_k] = N\sum_{i=0}^{N-1} |\theta_{k+1,i+1} - \theta_{k+1,i}|^2. \tag{4.21}$$

For the entropy based measure of uniformity, consider a collection of points $\theta_{k,0}, \ldots, \theta_{k,N-1} \sim \mu_k$ on Γ_k that are distributed according to the density $\mu_k(\cdot)$ (not necessarily uniform). Then for any function f, it is

$$\frac{1}{N}\sum_{i=0}^{N-1} f(\theta_{k,i}) \simeq \int_0^{2\pi} f(\theta)\mu_k(\theta)\mathrm{d}\theta,$$

yielding

$$S_k[\phi_k] \simeq S_k[\theta_{k+1}|\theta_k] = \frac{1}{N}\sum_{i=0}^{N-1} \log(\theta_{k+1,i+1} - \theta_{k+1,i}).$$

In this discrete virtual marker approximation, the local dilation rate at $\theta_{i,k}$, also called the local dispersion, reads

$$\mathrm{LD}_{k,i} = \frac{1}{\delta t}\log\frac{\theta_{k+1,i+1} - \theta_{k+1,i}}{\theta_{k,i+1} - \theta_{k,i}}. \tag{4.22}$$

Please note that for infinitesimally small time steps another formulation of this rate is given by Eq (4.6). Especially for level set methods, this formula is more applicable. Finally, the discrete optimization problem is given by

$$\phi_{k,\lambda} = \underset{\phi_k}{\operatorname{argmin}}\ F_k[\phi_k(\theta_k)|\theta_k] + \lambda U_k[\phi_k(\theta_k)|\theta_k], \quad \lambda > 0. \tag{4.23}$$

Note that in the discrete optimization problem we do not pose any requirements on the space of transformations ϕ_k ensuring condition Eq (4.8). If $\phi_k(\theta_{k,i+1}) - \phi_k(\theta_{k,i})$ $= \theta_{k+1,i+1} - \theta_{k+1,i} \leq 0$ for some k, we say that ϕ_k exhibits mapping violations.

Marker re-initialization for weakly-regularized contour flows.

In general, the distribution of virtual markers $\theta_k \sim \mu_k$ on Γ_k depends on the initial distribution of virtual markers $\theta_0 \sim \mu_0$ on Γ_0, since the density μ_k results from the transport of μ_0 by the flow. As a consequence, functions derived from the local contour flow like, e.g., the local dilation rate along Γ_k, may depend on the initial distribution on Γ_0. By re-initializing the distribution of virtual markers on Γ_k for any $k = 1, \ldots, K-1$, this dependence may be removed. A natural choice is to re-initialize with the uniform distribution, i.e., using

$$\theta_{k+1,i} = \phi_k(\xi_i) \tag{4.24}$$

with $\xi_i = 2\pi i$ for $i = 0, \ldots, N-1$ instead of Eq (4.19). To approximate the local flow $\phi = (\phi_k)_{k=0,\ldots,K-1}$ between contours, we thus solved the re-initialized optimization problem

$$\phi_{k,\lambda} = \underset{\phi_k}{\operatorname{argmin}}\; F_k[\phi_k(\xi)|\xi] + \lambda U_k[\phi_k(\xi)|\xi], \quad \lambda > 0. \tag{4.25}$$

with $\xi = (\xi_i)_{i=0,\ldots,N-1}$. For the local dispersion, e.g., this resulted in

$$\mathrm{LD}_{k,i} = \frac{1}{\delta t} \log \frac{\phi_{k,\lambda}(\xi_{i+1}) - \phi_{k,\lambda}(\xi_i)}{\xi_{i+1} - \xi_i} \tag{4.26}$$

with no dependence on the initial distribution of markers on Γ_0.

4.2.5 Algorithmic workflow

We summarize the proposed numerical workflow:

1. Given the segmented contours $\gamma_0, \ldots, \gamma_{K-1}$ as in Eq (4.1), determine the continuous representations Φ_k defined in Eq (4.2) satisfying the conditions Eq (4.3).
2. Determine contour based quantities like the curvature in Eq (4.4) or the center of mass in Eq (4.5).
3. To determine the (strongly-regularized) coordinate flow, consider N equally spaced markers

   $$\theta_0 = (\theta_{0,i})_{i=0,\ldots,N-1} \qquad \theta_{0,i} = \xi_i = \frac{2\pi i}{N}$$

   on the initial contour and choose a large λ value. Iteratively solve the regularization problem in Eq (4.25) to determine the coordinate markers θ_{k+1} for Γ_{k+1} from the coordinate markers θ_k for Γ_k. Since both θ_{k+1} and θ_k are approximately equally spaces, solving the minimization problem amount to choosing $\theta_{k+1,0}$.
4. To determine the (weakly regularized) local flow $\phi = (\phi_k)$ between successive contours, choose a small λ value and solve the regularization problems in Eq (4.25) to determine the coordinates $\theta_{k+1,i} = \phi_{k,\lambda}(\xi_i)$ on Γ_{k+1} based on N equally spaced markers $\xi_0, \ldots, \xi_{N-1}$ on Γ_k.

5. Determine contour flow based quantities like the local dispersion in Eq (4.26) or the local motion:

$$\mathrm{LM}_{k,i} = \|V_k(\xi_i)\| = \frac{\|\Phi_{k+1}(\phi_{k,\lambda}(\xi_i)) - \Phi_k(\xi_i)\|}{\delta t}, \tag{4.27}$$

see also Eq (4.9).

6. Map all quantities, based on contour flow as well as contour only, onto the (strongly regularized) global flow.

Numerical implementation and reproducibility.

All methods presented in this article are fully accessible as an open source Python-based toolbox `AmoePy` [DS3]. Implementations of the above methods and additional routines necessary to reproduce the figures and results of this article are part of this toolbox. Furthermore, `AmoePy` contains:

- An object-oriented analysis tool to handle cell contours, i.e., to shorten, extend, or manipulate existing data sets and to extract additional geometric quantities (e.g. area, perimeter, and normal vectors along the cell contour),
- A python class to perform Gaussian process regressions in 2d (space) and 3d (space-time) with different selectable kernel functions,
- Multiple testing routines based on artificial test data and experimental data, see Section A.1 and File A.1 for more details,
- Several routines generating videos of cell tracks with corresponding kymographs and occurring expansion/contraction patterns,
- A detailed documentation generated by the Python documentation tool Sphinx,
- A graphical user interface able to compute and present kymographs

`AmoePy`, as well as its graphical user interface, is updated regularly. Future outcomes of our ongoing research regarding for example cell segmentation routines and a forward model to simulate amoeboid cell motility will be also added to `AmoePy`.

The algorithm starts with initializing equally spaced markers, representing the starting points of the flow (see also the algorithmic workflow). Subsequently, the optimization problem in Eq (4.25) is solved contour-wise by gradient descent. Here, it is beneficial that the GPR also provides these gradients as a by-product, which decreases the computation time drastically; see Section A.1 and Fig A.4 for more details. for more details. Fig 4.2 shows the pseudocode to compute a regularized flow for a given regularization parameter λ. Being able to determine the regularized flow for a given parameter λ is the prerequisite to finally compute the global (strongly regularized) and local (weakly regularized) flows. These are based on the two regularization parameters $\lambda_{\text{glo}} \gg \lambda_{\text{loc}} > 0$. The local quantities are then represented in the coordinate system of the global flow, yielding the kymograph representation. A pseudocode describing the computation of these kymographs is shown in Fig 4.3.

In contrast to level set and electrostatic methods, the optimization approach presented here is not relying on the computation of intermediate field lines or contours

```
Algorithm 1: RegFlow
  Input: $\gamma_0, \dots, \gamma_{K-1},\ r \in (0,1),\ \sigma_n > 0,\ N \in \mathbb{N},\ \lambda \geq 0$
  Output: $\Gamma_0, \dots, \Gamma_{K-1},\ \boldsymbol{\theta}_0^{\text{eval}}, \dots, \boldsymbol{\theta}_{K-1}^{\text{eval}}$

  /* Initialize evenly distributed starting points                */
  $\boldsymbol{\theta}_{\text{init}} = \texttt{NormalizedArcLengthCoordinates}(\gamma_0)$;
  $\boldsymbol{\theta}_0^{\text{eval}} = \left[0,\ \frac{2\pi}{N},\ 2 \cdot \frac{2\pi}{N},\ \dots,\ (N-1) \cdot \frac{2\pi}{N}\right]$;
  $\Gamma_0 = \texttt{GPR}(\gamma_0,\ \boldsymbol{\theta}_{\text{init}},\ \boldsymbol{\theta}_0^{\text{eval}},\ r,\ \sigma_n)$;

  /* Compute virtual marker flow contour-wise                     */
  for $k = 1$ to $K-1$ do
      $\boldsymbol{\theta}_{\text{init}} = \texttt{NormalizedArcLengthCoordinates}(\gamma_k)$;
      $\boldsymbol{\theta}_k^{\text{eval}} = \texttt{MinimizeFunctional}(\Gamma_{k-1},\ \gamma_k,\ r,\ \sigma_n,\ \lambda)$;
      $\Gamma_k = \texttt{GPR}(\gamma_k,\ \boldsymbol{\theta}_{\text{init}},\ \boldsymbol{\theta}_k^{\text{eval}},\ r,\ \sigma_n)$;
  end
  return $\Gamma_0, \dots, \Gamma_{K-1},\ \boldsymbol{\theta}_0^{\text{eval}}, \dots, \boldsymbol{\theta}_{K-1}^{\text{eval}}$
```

Figure 4.2. Algorithm to compute regularized flows. The algorithm input consists of three parameters r, σ_{noise}, and N regarding the Gaussian process regression, a regularization parameter λ and the initial (segmented) contours $\gamma_0, \dots, \gamma_{K-1}$. The regularized flow is described by the output variables $\Gamma_0, \dots, \Gamma_{K-1}$ denoting smooth contours evaluated at a finite number of coordinate markers $\boldsymbol{\theta}_0^{\text{eval}}, \dots, \boldsymbol{\theta}_{K-1}^{\text{eval}}$.

for sufficiently small grid sizes. Notably, level set methods are computationally more expensive because of the time integration of virtual markers along these field lines[29, 28]. On the other hand, our algorithm contains additional steps such as the re-initialization of virtual markers via Gaussian process regression and the computation of kymograph quantities which may lead to slower computation times than other empirical mapping algorithms. A direct comparison of computation times of these methods is rather difficult since some algorithms rely on a varying number of virtual markers or a more image-based implementation (e.g. in `ImageJ`). Moreover, the computation time depends on varying configurations of these methods, e.g., the resolution of the underlying grid in LSM, or the number of iterations/tolerance parameters used in optimization approaches such as ours. Unfortunately, a "biological true" mapping does not exist, with which the accuracy of each of those methods can be measured.

Nevertheless, we provide the computation times of our algorithm "RegFlow" shown in Fig 4.2 on a standard computer, see Fig A.5. For a cell track of 500 contours and 400 virtual markers, the computation time for generating the mapping between two successive contours was measured for different regularization parameters λ. Not surprisingly, the computation time decreases for the end-member cases $\lambda \to 0$ and $\lambda \gg 0$, where the underlying functional is only defined by one leading term. For each choice of λ, the median of the computation time is below $2s$ where most of the cases required sub-second computation times. Furthermore, no mapping violations occurred during the cell track despite larger shape deformations and membrane spikes. Therefore, our method has the potential to be used specifically for cell motility models that rely on a fast computation of stable virtual marker trajectories.

Algorithm 2: ComputeKymographs

```
Input: $\gamma_0, \dots, \gamma_{K-1},\ r \in (0,1),\ \sigma_n > 0,\ N \in \mathbb{N},\ \lambda_{\text{glo}} \gg \lambda_{\text{loc}} \geq 0$
Output: $\Gamma_0, \dots, \Gamma_{K-1},\ C_0, \dots, C_{K-1},\ \text{LM}_0, \dots, \text{LM}_{K-2},\ \text{LD}_0, \dots, \text{LD}_{K-2}$

/* Initializing global flow                                          */
$\Gamma_0^{\text{glo}}, \dots, \Gamma_{K-1}^{\text{glo}},\ \boldsymbol{\theta}_0^{\text{glo}}, \dots, \boldsymbol{\theta}_{K-1}^{\text{glo}} =$ RegFlow($\gamma_0, \dots, \gamma_{K-1},\ r,\ \sigma_n,\ N,\ \lambda_{\text{glo}}$);

/* Computing curvature at global flow                                */
$C_0, \dots, C_{K-1} =$ Curvature($\Gamma_0^{\text{glo}}, \dots, \Gamma_{K-1}^{\text{glo}},\ r,\ \sigma_n$);

for $k = 1$ to $K-1$ do
    /* Computing dynamic quantities between two contours             */
    $\Gamma_{\text{start}}^{\text{loc}},\ \Gamma_{\text{end}}^{\text{loc}},\ \boldsymbol{\theta}_{\text{start}}^{\text{loc}},\ \boldsymbol{\theta}_{\text{end}}^{\text{loc}} =$ RegFlow($\gamma_{k-1},\ \gamma_k,\ r,\ \sigma_n,\ N,\ \lambda_{\text{loc}}$);

    $\text{LM}_{k-1}^{\text{loc}} =$ LocalMotion($\Gamma_{\text{start}}^{\text{loc}},\ \Gamma_{\text{end}}^{\text{loc}}$);
    $\text{LD}_{k-1}^{\text{loc}} =$ LocalDispersion($\boldsymbol{\theta}_{\text{start}}^{\text{loc}},\ \boldsymbol{\theta}_{\text{end}}^{\text{loc}}$);

    /* Computing quantities along global flow                        */
    $\boldsymbol{\theta}_{\text{eval}} = \boldsymbol{\theta}_{k-1}^{\text{glo}}$;
    $\text{LM}_{k-1} =$ GPR($\text{LM}_{k-1}^{\text{loc}},\ \boldsymbol{\theta}_{\text{start}}^{\text{loc}},\ \boldsymbol{\theta}_{\text{eval}},\ r,\ \sigma_n$);
    $\text{LD}_{k-1} =$ GPR($\text{LD}_{k-1}^{\text{loc}},\ \boldsymbol{\theta}_{\text{start}}^{\text{loc}},\ \boldsymbol{\theta}_{\text{eval}},\ r,\ \sigma_n$);
end

return $\Gamma_0, \dots, \Gamma_{K-1},\ C_0, \dots, C_{K-1},\ \text{LM}_0, \dots, \text{LM}_{K-2},\ \text{LD}_0, \dots, \text{LD}_{K-2}$
```

Figure 4.3. **Algorithm to compute curvature, local motion and local dispersion.** The two regularization parameters λ_{glo} and λ_{loc} are used in the RegFlow routine (see Fig 4.2) to determine the global (strongly regularized) and local (weakly regularized) flow. The output comprises the smoothed contours $\Gamma_0, \dots, \Gamma_{K-1}$ as well as geometric quantities of interest such as curvature, local motion, and local dispersion.

4.3 Analysis of amoeboid cell motility by means of regularized contour flows

4.3.1 Degree of regularization controls distribution of virtual markers

We applied our approach to time-lapse microscopy data of the social amoeba *D. discoideum*. Fig 4.4 illustrates the data acquisition process. For each time point t_k, we obtained a sequence of segmentation points from a fluorescence image (see panel (A) and Eq (4.1)) and their corresponding continuous representation Φ_k (see panel (B) and Eq (4.2)). The entire sequence of continuous contours is shown in panel (C), while the trace of the cell track and the center of mass trajectory are shown in panel (D).

In Fig A.6, estimations of continuous representations for different hyperparameters are shown. In contrast to Fig 4.4, the underlying snake of segmented points was obtained from noisy fluorescence image data resulting in many abrupt changes of the contour's direction. In this context, the curvature for different contour estimates is displayed, highlighting the effect of underfitting and overfitting. Notably, the GPR provides an automated way of estimating the hyperparameters balancing the model's complexity and its error residuals, see Section 3.1.4 for more information. This way,

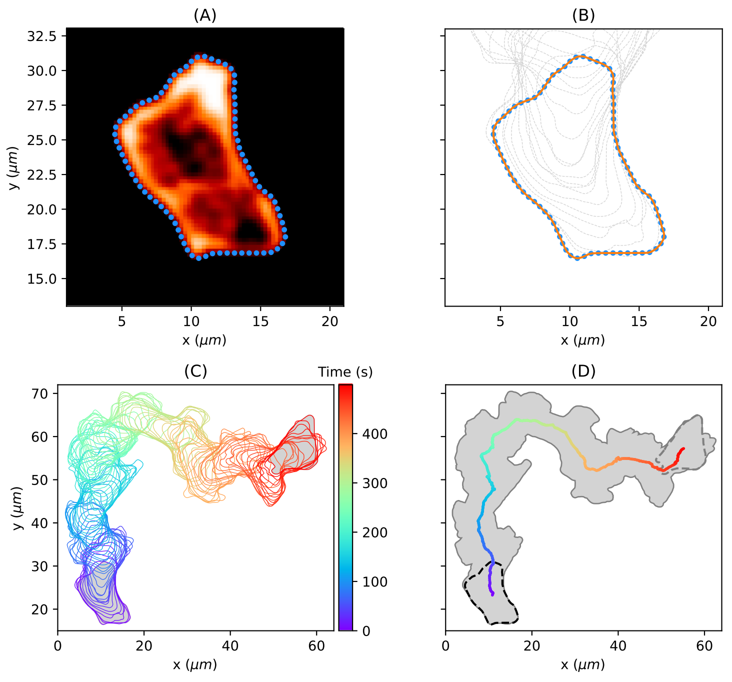


Figure 4.4. Cell trajectory of a persistently moving amoeba. (A) Fluorescence image with closed string of $M = 400$ equidistant nodes resulting from the segmentation process; shown is only every fifth segmentation point (blue points). **(B)** Smooth representation Φ_k of the cell contour (orange line) obtained by spatial Gaussian regression on the segmentation points. Every fifth cell contour is displayed as dashed gray line. **(C)** Entire cell track of $K = 500$ cell contours (only every fifth shown). **(D)** Global trace of the cell track (gray area) and the trajectory of the center of mass of the contour (solid line, color coded as in panel (C)). The initial contour is shown as dashed black line and the final contour as dashed gray line.

an accurate approximation of the cell contour can be obtained even for noisy data while preserving the main characteristics of its shape.

In Fig A.7, the effect of different imaging frequencies on the resulting kymographs is shown. In this context, local motion kymographs are computed by using (i) the entire microscopy data (one image/contour per second) and (ii) down-sampled data sets based on every 2nd, 3rd, 5th, and 10th image/contour. While global characteristics of the cell track are captured even for a lower temporal resolution ($\delta t > 3$), the identification of local membrane changes becomes impossible. We therefore recorded each cell track with an imaging rate of one frame per second ($\delta t = 1$). See also Fig A.3, in which contour mappings for larger shape deformations are displayed. Here, the underlying cell track was recorded with an imaging rate of $\delta t \approx 3.13s$.

The continuous representations $\Phi_0, \ldots, \Phi_{K-1}$ of all contours are the input to the optimization problem Eq (4.16) to determine the regularized contour flow. Fig 4.5 shows the impact of the regularization parameter λ on the virtual marker trajectories (shown for two illustrative contours). In the absence of any regularization (panel (A), $\lambda = 0$), virtual markers are thinned out in some regions (linked to expanding

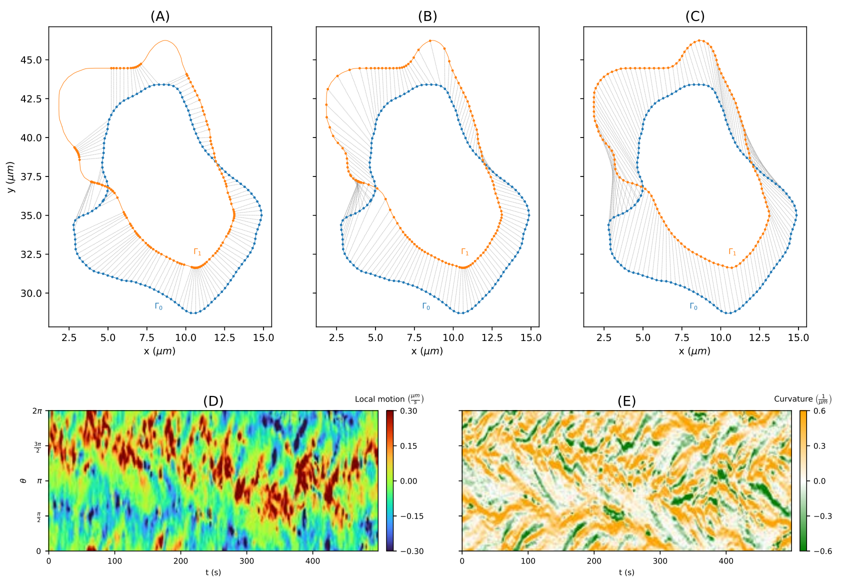


Figure 4.5. Impact of regularization on the distribution of virtual markers for **(A)** no regularization ($\lambda = 0$), **(B)** weak regularization with $\lambda = 0.1$, and **(C)** strong regularization with $\lambda = 1000$, illustrated on two frames (roughly 20s apart for illustration purpose). Using the strongly regularised so-called coordinate markers as a means to map local characteristics into a kymograph, the lower panel shows the local motion **(D)** and curvature**(E)**. The local motion is defined by the magnitude of each mapping vector, which are determined based on the weakly regularised marker flow. All panels correspond to the persistently motile cell of Fig 4.4.

areas), while they are clustered in others (in particular at the back of the cell). The case $\lambda = 0$ corresponds to minimizing the translation from the first contour to the next, i.e., each virtual marker on the first contour is linked to its nearest neighbor on the second contour. This may result in mapping violations $\theta_{k+1,i+1} - \theta_{k+1,i} \leq 0$ on the second contour. An example of mapping violations can be seen on the left-hand side in panel (A). As mentioned earlier, the flow obtained with $\lambda = 0$ is defined by the shortest path flow or equally the reversed normal flow. Thus, instead of taking the trajectories to the nearest neighbors on the next contour, one could also choose the shortest normal vectors from the second contour to the first one.

In the weakly regularized case, virtual marker thinning and clustering is still prominent (see panel (B) with $\lambda = 0.1$), but to a lesser extent. Moreover, in the presence of regularization, virtual marker trajectories are interdependent, which results in trajectories without mapping violations. In the limit of strong regularization, the marker points remain uniformly distributed on every contour, while minimizing the overall distance between contours (see panel (C) with $\lambda = 1000$). This makes the strongly regularized virtual marker trajectories an ideal candidate for a time-evolving reference frame and corresponds to the previously defined MCCS coordinate system.

By choosing a strong regularization for the coordinate system, however, information on local contour changes is largely lost. Therefore, we used the strongly regularized case only to determine the coordinate system (set of $N = 400$ virtual marker trajectories), while we determined local contour characteristics (e.g. local motion or local dispersion) based on a re-initialized weakly regularized flow. The local characteristics were subsequently represented in the coordinate system obtained from a strongly regularized flow. In panel (D), the local motion obtained from the re-initialized weakly regularized flow with $\lambda = 0.1$ is shown for the same cell track as in Fig 4.4 and the entire time-lapse microscopy recording (500 frames, $\delta t = 1$s). The local motion clearly shows regions of fast-moving membrane parts at the leading edge (red areas) and at the back of the cell (blue areas). The curvature kymograph in panel (E) shows characteristic lines of strongly convex (orange) and concave (green) membrane parts. Inclined lines of curvature (and local motion) may result from adherent parts of the cell moving along the cell contour as well as shifting effects of virtual markers due to arc length changes. It is important to notice that a kymograph depends on the underlying time-evolving coordinate system.

In contrast to the procedure mentioned above, one may also compute local characteristics along the global flow without re-initializing from a uniform distribution of markers after every time step. However, this is not recommended, because either local information gets lost (strongly regularized flow) or clustering and thinning effects of markers become too prominent (weakly regularized flow); see Fig A.8 for global flows based on different regularization parameters and the resulting kymographs.

As shown in Fig A.9, we further challenged our algorithm by computing a strongly regularized (global) flow of the cell track from Fig 4.4 based on a few frames only (8 out of 500). Even under such extreme conditions, the algorithm produced stable virtual marker trajectories, i.e., trajectories without mapping violations.

4.3.2 Kymographs of local properties show characteristic patterns of amoeboid motility

The kymographs of local dispersion (LD), local motion, and curvature can be used to visualize, analyze, and quantify the expanding activity of a cell track. Note that the three quantities are closely related. Fig 4.6 shows cell tracks and corresponding kymographs for three different motility patterns: (A) persistently motile cell; (B) weakly motile cell; and (C) an almost stationary cell. All kymographs are smoothed by a Gaussian filter with a standard deviation of three markers in space and a standard deviation of one contour in time. See Fig A.10 for the same kymographs but without smoothing. Moreover, videos of all three cell tracks and their corresponding kymographs can be found in Video A.1–Video A.3.

For the persistently motile cell, the LD kymograph shows strong (positive) activity in a band-like structure along roughly half of the cell contour (width π), while the activity of local dispersion is less localized for the weakly motile cell and much less pronounced for the stationary cell. A similar scenario is seen in the local motion kymographs of the three cells. In broad terms, the local motion kymographs show

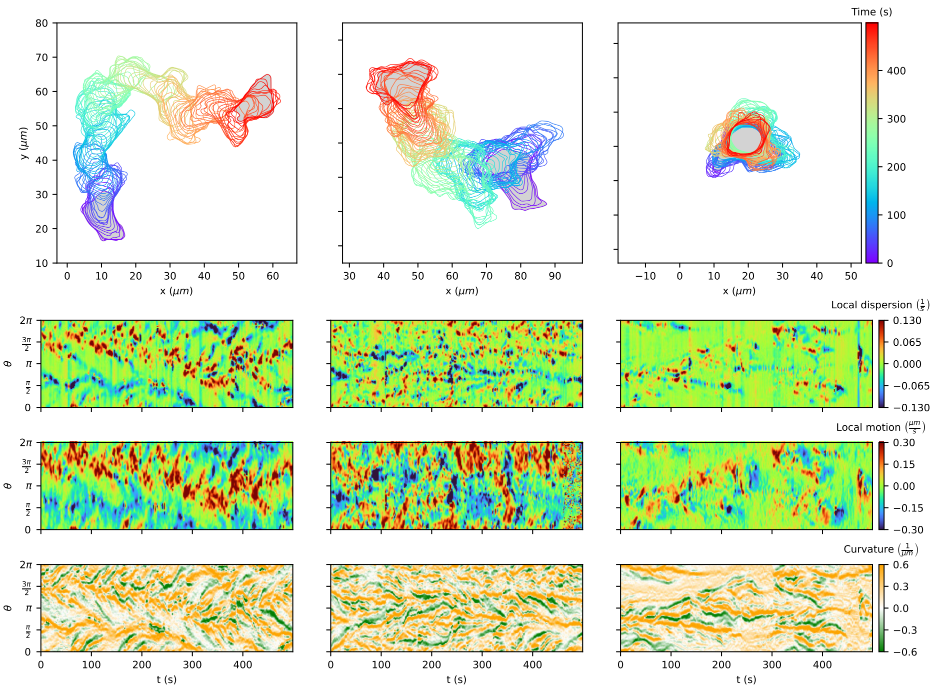


Figure 4.6. Comparison of different cell tracks of *Dictyostelium discoideum*: persistently motile (left), weakly motile (middle) and almost stationary (right). The corresponding kymographs contain information on the local dispersion (left), local motion (middle) and curvature (right). For details see text.

more activity, e.g., areas of red and dark red color, than the LD kymographs. This is also apparent from a correlation plot of the two quantities (see Fig A.11).

In the following section, we chose the local dispersion as a basis to identify expansions being a product of local velocity and curvature. Another reason to choose LD as the quantity to define expansions is that many patches of high activity in the local motion kymograph contain multiple LD areas. The LD allowed us to divide these patches of high local motion into single separated expansions with high LD rate.

4.3.3 Virtual marker dispersion allows to identify and characterize expansions

Using the persistently motile cell track in Fig 4.4, we describe next, how to use the LD to define expansion areas and expanding events. Based on the local dispersion kymograph in Fig 4.7A, we defined areas of medium (light red) and high (dark red) expanding activity as well as medium and high contracting activity (light and dark blue, respectively), see Panel (B). In this context, we determine thresholds for expanding activity by dividing the 90th percentile of all positive LD values of the kymograph into three intervals of equal length. For contracting activities ($LD < 0$) the opposite thresholds were taken. By leaving out the largest and smallest values of the LD kymograph, the classification becomes less dependent on outliers. Additionally, we performed a prior smoothing of the kymograph as mentioned in the

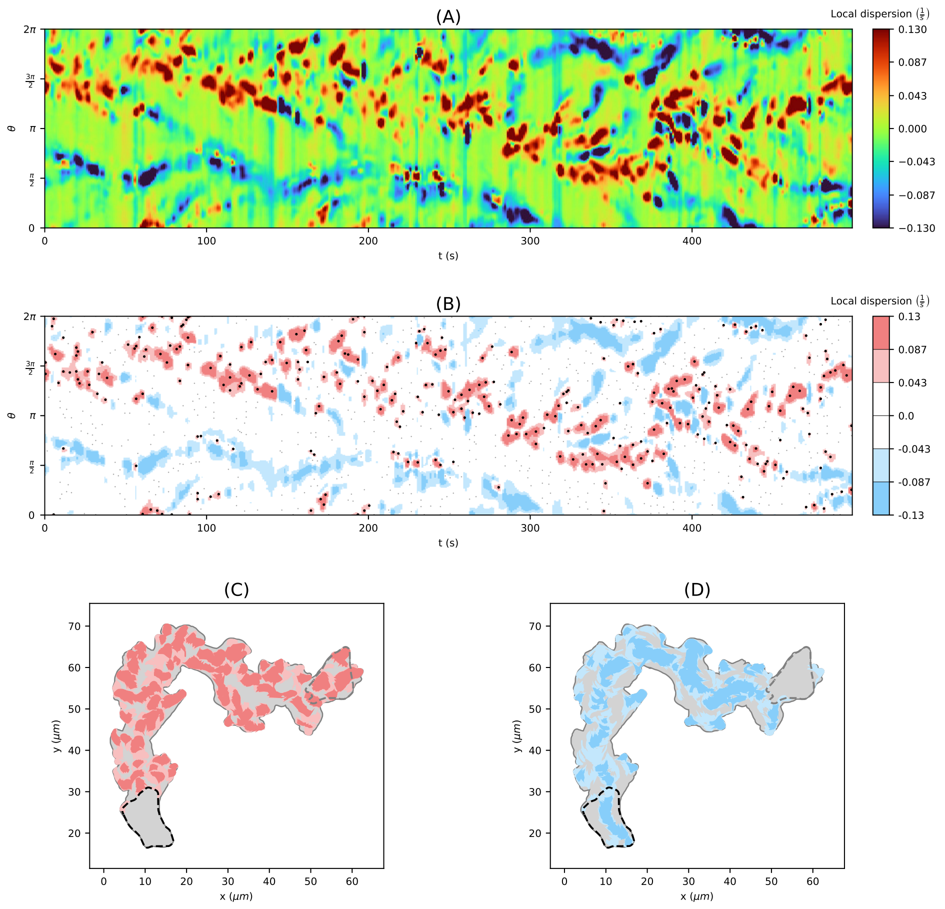


Figure 4.7. From the local dispersion kymograph to expanding areas and expansion events. (A) Local dispersion of a persistently motile cell as in Figs 4.4 and 4.6 (left-hand side). **(B)** Discretised local dispersion with different areas of activity: high (dark red), medium (light red), low (white) expanding activity, and low (white), medium (light blue), high (dark blue) contracting activity. Local maxima of positive local dispersion are depicted as black dots. Areas of medium and high expanding **(C)** and contracting **(D)** activity mapped back on the trace of the cell track.

first paragraph of the previous section to reduce noise and, therefore, to reduce the number of small and separated patterns.

To highlight the events of the highest local expanding activity, we included positive local maxima of the LD kymograph (black dots) in panel (B). Local maxima falling inside regions with high or middle expanding activity were depicted as bold dots. Using the time-evolving coordinate system obtained from strongly regularized flow, the expansion/contraction areas shown in panel (B) are mapped back into the 2d plane of amoeboid motion, see panels (C) and (D), respectively. As a result, we obtain an automated visualization of the expanding activity during amoeboid locomotion.

The kymograph in panel (B) clearly shows the trace of expanding activity at the leading edge, located initially at around $3\pi/2$ and then shifting towards π. At the same time, contracting activity occurs mainly at a distance of π from the leading

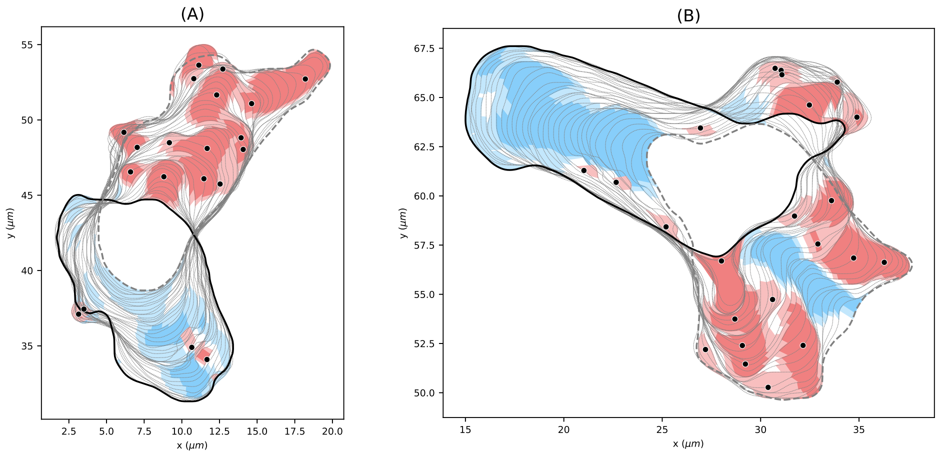


Figure 4.8. Expanding and contracting areas with corresponding expansion events. Illustrative sequence of the contour dynamics for $96s \leq t \leq 144s$ (left), and $316s \leq t \leq 350s$ (right) based on the cell track shown in Fig 4.7. Features with high and medium expanding activities are shown in dark and light red, respectively. Features with high and medium contracting activities are shown in dark and light blue, respectively. All patterns shown possess a minimal growth time of $3s$. The black dots show local maxima of the local dispersion in areas of medium and high expanding activity.

edge. In panels (C) and (D), one can nicely see the explorative dynamics of the expansions at the cell front and the stably retracting back of the cell, the so-called uropod, where dark blue areas indicate faster contractions. Analogous graphics for a collection of motile and stationary cells can be found in File A.2 and File A.3, each containing 12 cell tracks.

In Fig 4.8, we present a close-up of two sequences of cell contours. The core expanding areas that shape the evolving cell contour can be seen clearly, e.g., in panel (A). The events of the highest expanding activity (black dots) seem to drive the expansion in many cases. Panel (B) illustrates strong contracting activity at the uropod, and the contraction of the membrane between two nearby expansions (blue area sandwiched between red areas). The opposite effect can be seen in panel (A) at the back of the cell, where a concave region between two convex contractions is identified as expansion. These patterns nicely illustrate the concept of local dispersion, being a product of the local velocity of virtual markers and the curvature along the contour segment.

In Fig A.12, every identified pattern with minimal growth time $\Delta t \geq 3$ is shown for the underlying cell track.

4.3.4 Statistical analysis of motility patterns

In this section, we illustrate the ability to statistically analyze a cell track based on our regularized contour flow approach. We used the persistently motile cell track for illustration.

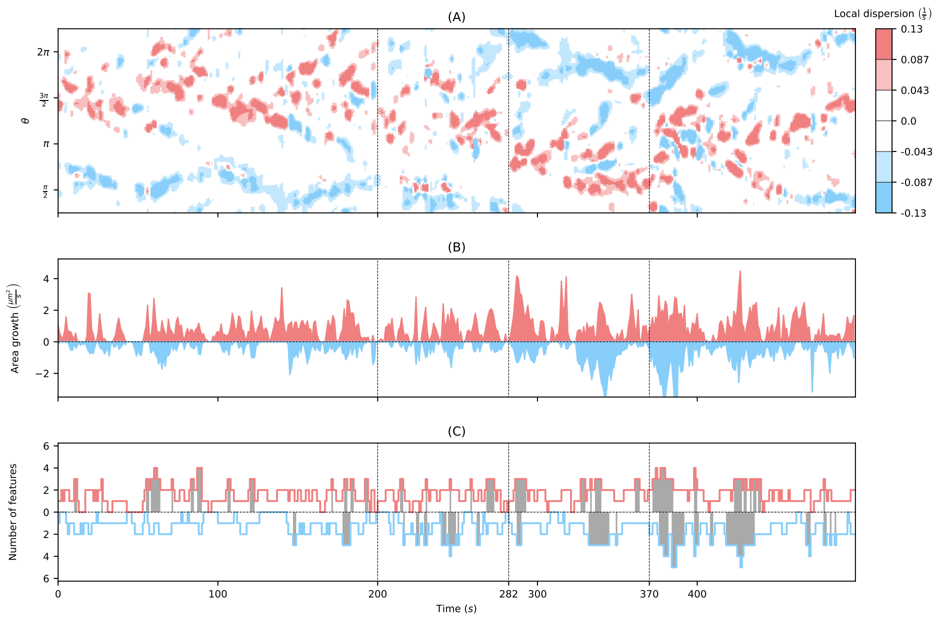


Figure 4.9. Statistical analysis of example cell track. (A) Local dispersion kymograph with thresholds as in Fig 4.7. **(B)** Area growth of cell segments which are part of identified expansions (red) and contractions (blue) of high and middle intensity. **(C)** Number of expanding and contracting areas with high intensity with respect to time. The time with > 2 features are colored gray to highlight the change in activity in the different phases.

Fig 4.9A shows the local dispersion kymograph of the persistently motile cell divided into four different phases (note that here the y axis starts at $\pi/4$). Until $t = 200s$, the cell moves upward with well observable expansions roughly between 1.1π to 0.2π, while the uropod is slightly above $\pi/2$. Then, the cell begins a phase of reorientation that lasts until $t \approx 280s$, where larger expansions occur also at the former back of the cell. Subsequently, the cell changes its direction downward toward $3/4\pi$ to π (in the video it moves rightwards). In the last phase, the cell moves to the right-hand side by creating expansions at the front left, front right, and again front left of the cell. See Video A.1 for a better understanding of the cell track.

We analyzed the expanding and contracting areas shown in Fig 4.7C and 4.7D with respect to the activity level (medium/high), duration, and position along the cell contour. In addition, we investigated the differences between expansions and contractions. Fig 4.9B displays the area growth of expansions and contractions for high activity. The identified expanding and contracting areas are naturally partitioning by the sequence of contours into smaller 'slices' (see, e.g., Fig 4.8). We defined the area growth of an expansion/contraction as the area of the slice divided by the frame rate δt. We observed that the overall change of cell area attributed to expansions of high activity is substantially larger than for contractions of high activity. This illustrates that the cell motility of this cell track is driven by a higher number of fast (and potentially explorative) expansions, and a small number of fast contractions. Since in broad terms, the total area gain balances the total area loss, it further illustrates that area loss is to a larger extent attributed to slower and steadier contractions than it is for expansions. After the second phase ($t > 280$), when the cell has completed reorientation, the area change for both, expansions and contractions increased and even more so at the beginning of the last phase ($t > 370$).

Finally, in panel (C), the number of simultaneous expansions and contractions is shown (high activity only). In the first phase of the upward moving cell, the number of expansions is much higher than the number of contractions (mean: 1.52 vs. 1.12). In addition, we determined the fraction of time $fT_{\text{exp}>2}$ and $fT_{\text{contr}>2}$ with more than two simultaneous expansions and contractions, respectively. In the first phase, it is $fT_{\text{exp}>2} = 0.146$ vs. $fT_{\text{contr}>2} = 0.035$. In the following phases, these fractions increase for contractions: $fT_{\text{contr}>2} = 0.198; 0.195; 0.341$ (for phase $2; 3; 4$), whereas for expansions the fraction is slightly smaller in the second phase and larger in the last two phases: $fT_{\text{exp}>2} = 0.123; 0.207; 0.217$ (for phase $2; 3; 4$). This illustrates that the increased activity, as seen in panel (B), goes along with an increased number of expansions and contractions. Moreover, this indicates a more explorative character of the cell motion in phases three and four, while the cell seemed to be more stabilized during the first phase.

Fig 4.10 gives further insight into the expanding activity. Panel (A) shows the distribution of LD values of all virtual markers within expanding and contracting areas with high activity. In other words, the local dispersion distribution shows only values that are larger than the thresholds for high expanding activity or smaller than the threshold for high contracting activity (the thresholds are ± 0.087). The distribution of the LD of expansions further extends towards large values than the corresponding distribution of contraction towards small values (median of 0.13/s vs. -0.12/s).

In panels (B) and (C), the distributions of local motion and the curvature are shown for all virtual markers within expanding and contracting areas of high activity (same areas as above). For the local motion, we observed major peaks around $0.29\,\mu m/s$ and $-0.21\,\mu m/s$ for expansions and contractions, respectively. Minor peaks correspond to inward expansions and outward contractions discussed in the previous section and shown in Fig 4.8 (see red expanding areas within a blue contracting region in panel (A), and a blue contracting area within an expanding region in panel (B)). In line with this observation, minor peaks of concave (negative) curvature and major peaks for convex (positive) curvature are shown.

Fig 4.10D shows the distribution of high activity expansions in direction of the moving cell. We identified two peaks in direction of the cell movement (front-left and front-right). Another peak is located at the back of the cell. A similar behavior was presented for pseudopods in [18, 6], where two different types of pseudopods were distinguished: (left/right) splitting pseudopods and *de-novo* pseudopods. By comparing the correlation between the growth time and the area of expansions and contractions in panel (E), we observed that expansions possess an area often twice as large as contractions with similar growth times. This indicates the difference between the faster and more explorative character of expansions at the cell front

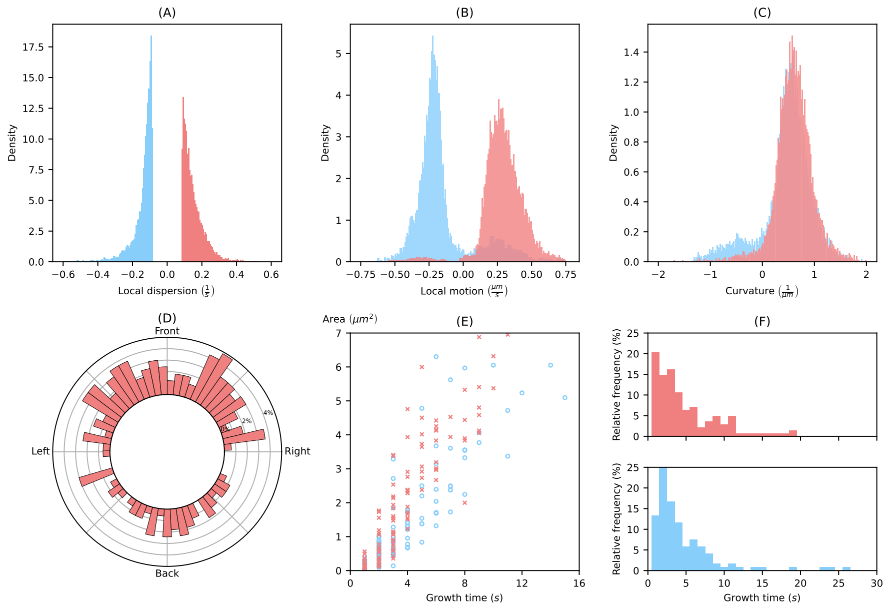


Figure 4.10. Statistical analysis of example cell track. Distributions of local dispersion **(A)**, local motion **(B)** and curvature **(C)** inside expanding and contracting patterns with high intensity. **(D)** Circular histogram displaying the angle, where high expanding activity appears along the cell contour. **(E)** Correlation between area and growth time of identified patterns. **(F)** Histograms of growth times of expansions and contractions with high intensity.

and the slower and stably retracting character of the uropod. This is also in line with the corresponding activities shown in Fig 4.9B.

Finally, in panel (F) the distribution of growth times is shown for both, high activity expansions and contractions. Note that we used the term 'growth time' for both, expansions as well as contractions, owing to the fact that also contracting areas can be moving in an outward direction, as discussed earlier. The majority of growth times fall inside a range of $0s$ to $10s$. Nevertheless, there are patterns, especially long persistent uropods, with growth times much larger than the range presented in these histograms. The average growth time of pseudopods observed in [6] is much higher ($12.8\,s$) than the growth times of expansions presented in this work ($4.9\,s$). This is not surprising, since our definition of expansions also takes short-lived objects into account. For example, the average number of expansions per minute for the persistently motile cell track (≈ 17.0) is much higher than the average frequency of pseudopods per minute (2.9 ± 0.2) as observed in [6]. Additionally, concave regions between two contractions as in Fig 4.8A were detected as expanding areas as well, which raises the total number of expansions. See File A.2 and File A.3 for similar graphics of a collection of 24 cell tracks, also containing the other two tracks from Fig 4.6.

4.3.5 Application to other kinds of cell motility

In this section, we demonstrate the flexibility of our method to study other kinds of cell motility. To this end, we extracted sequences of cell contours from videos of early

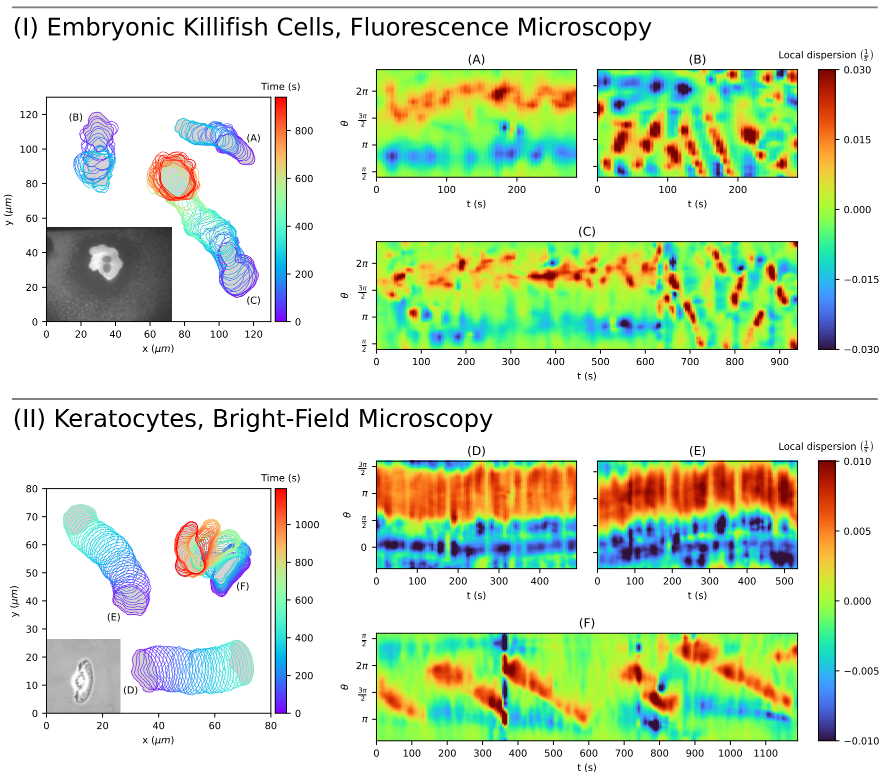


Figure 4.11. Application to other kinds of cell motility. Displayed are three tracks for embryonic killifish cells **(A-C)** and keratocytes **(D-E)**, and their corresponding local dispersion kymographs. Protrusive migration can be seen in **(A, C)**, circular waves around the cell border in **(B, C, F)**, and a steady persistent translation of the contour in **(D, E)**.

embryonic killifish cells (*Fundulus heteroclitus*) [107, 108, 109] and keratocytes cultured from Central American cichlids (*Hypsophrys nicaraguensis*) [21, S1-S3 Movies]. While the images of the embryonic killifish cells were obtained from fluorescence microscopy, bright-field microscopy was used to record the keratocytes. As before, the images were segmented by using a modified version of the active contour (snake) algorithm described in [24, 19].

In Fig 4.11, three cell tracks for both applications, embryonic killifish cells and keratocytes, and corresponding local dispersion kymographs are shown. In panels (A) and (C), protrusion-driven cell motility can be observed for the embryonic killifish cells. Moreover, rotating waves around the leading edge (so-called circular movements) can be seen in panel (B) and in the last third of panel (C) as diagonal lines. In the chosen parametrization, right upward diagonals indicate counterclockwise rotations of protrusions while right downward diagonals indicate clockwise rotations. Notably, the switching between these kinds of locomotion can be nicely seen in panel (C) at $t \approx 630$. In addition, we refer to the video accessible on [108] showing the cell track from panel (B) with dominant circular movements.

On the contrary, keratocytes possess a steady and persistent type of cell migration with only minor shape deformations. However, keratocytes can also show a more crawling kind of cell motility depending on the adhesion strength. In [21, S1-S3 Movies], cell track recordings for different adhesion strengths are provided: intermediate adhesion strength (panel (D)), low adhesion strength (panel (E)), and high adhesion strength (panel (F)). For low and intermediate adhesion strength, persistent cell track can be observed with a slightly higher local dispersion at the cell front for the case of low adhesion strength. For high adhesion strength, the cell migrates slower and exhibits larger cell shape deformations. Again, diagonal lines in panel (F) indicate circular movements as described earlier. Due to segmentation/mapping difficulties resulting from jumps in the video recording, pronounced artifacts can be seen as vertical lines at $t = 360s$ and $t = 750s$.

These test applications underline the flexibility of our algorithm, handling contours obtained from fluorescence images as well as bright-field recordings. Moreover, the underlying data was recorded with a lower temporal resolution, $5s$ per image compared to only $1s$ in our data. For the case of embryonic killifish cells, which possess local contour changes comparable to our data sets, a temporal resolution of $5s$ is relatively low. This can be nicely seen in panel (B). Since the movement of keratocytes is much slower with less prominent contour changes, kymographs with relatively high resolution can be acquired for a sufficient temporal resolution of $5s$. This shows the feasibility of our algorithm for different imaging frequencies, see also Fig A.7. Finally, the local dispersion kymographs of these test cases show distinct differences from previous kymographs regarding amoeboid cell migration. This indicates the potential of our algorithm for classifying different kinds of cell motility.

4.4 Discussion

With the ever-increasing amount of live cell imaging data and the continuously growing computational power, computer-automated techniques to analyze the morphology of cells have steadily developed over the past two decades. In particular, in cell motility research, morphological characteristics are commonly used to pinpoint phenotypic differences between mutant cell lines thus highlighting the mechanistic role of individual components of the underlying signaling pathways. While many static measures of cell shape have already been introduced early on [110, 111], dynamic measures that quantify the temporal evolution of the cell shape proved to be more difficult to implement. First attempts focused on temporal changes of the projected cell area to deduce overall protrusion rates, for an example see [112]. These approaches were later refined by local measures of cell boundary motion along selected line segments perpendicular to the cell border [113]. Also, local space-time plots have been defined in this way [114, 115]. However, in all these cases, the direction of interest or the local placement of the kymograph had to be chosen manually, which severely limits a reliable long-time tracking of more complex cell shapes and introduces an arbitrariness related to the manual processing.

The most promising approach to overcome this limitation relies on an active contour (snake), a closed chain of connected nodes (virtual markers) that is placed along the cell perimeter [116]. Different rules have been proposed to propagate the markers from one contour to the next. In some cases, the distance between virtual markers is kept constant and markers are added or removed as the contour evolves [117]. Other approaches that are inspired by mechanical spring models or concepts from electrostatics keep the number of markers constant and allow for local variations in the marker spacing [28]. Here, we can distinguish two limiting cases. On the one hand, equidistance is enforced and markers on adjacent contours are connected in a one-to-one mapping by minimizing the sum of square distances between pairs of connected markers [24]. On the other hand, markers are propagated from one contour to the next in a normal direction (normal flow) while the marker spacing evolves without constraints. The latter approach has been implemented using a level set method to cope with problems related to finite time sampling [29]. However, high computational costs and the rapid buildup of highly uneven marker distributions limit the use of the level set method in practical applications.

In this article, we have introduced a family of marker flows that incorporates these different cases into a general framework. In particular, the regularization that we introduced in Eq (4.16) includes the two extreme scenarios described above as limiting cases. In the limit of large λ, we obtain an equidistant mapping, whereas, in the limit of small λ, we approach the shortest path flow, respectively, the reverse normal flow. Tuning λ allows us to systematically shift between these two limits.

Once a flow of virtual markers is computed on the evolving cell contour by any of these methods, it defines a coordinate system, in which different local quantities can be displayed, such as curvature, membrane displacement, or the intensity of a fluorescently tagged membrane-associated protein. Essential to our approach is the clear separation of local quantities derived by weakly regularized flows between two consecutive contours and the coordinate system which is based on a single strongly regularized (global) flow onto which each quantity of interest is mapped. This way, trajectories of each quantity can be obtained for the entire time period. Note that for all of these coordinate choices it is generally acknowledged that the dynamics of the virtual membrane markers do not reflect the motion of specific membrane lipids or proteins, as the membrane itself is a very complex and dynamic structure [29, 24]. In particular, lateral flows may occur due to membrane recycling, so that the dynamics of individual molecules or domains in the membrane do not necessarily correspond to morphological changes and will prevent a one-to-one mapping of molecular markers on adjacent contours.

Amoeboid motion is primarily driven by localized membrane protrusions, so-called pseudopodia. Identifying and tracking pseudopodia has thus been an important focus of the morphodynamic analysis of amoeboid cells. The first substantial pseudopod statistics were generated by computer-assisted manual image processing, relying on the expert judgment of the investigator [20, 6]. From this, a first automated software routine for pseudopod tracking was developed [23] and successfully applied also to analyze the chemotactic navigation of amoeboid cells [18]. It relies on a complex decision tree that defines pseudopodia based on a sequence of threshold criteria applied to the local curvature, the virtual marker movement, and the local area change. In this way, the frequency and direction of pseudopod formation, their sizes, lifetimes, and other quantitative measures were extracted. While successfully providing a first quantitative database of pseudopod characteristics, this approach has the drawback that it requires the choice of several parameters that are tuned to the characteristic properties of pseudopods in starvation-developed *D. discoideum* cells. If cells display protrusions with a more diverse range of shapes and time scales, a reliable tracking is difficult to achieve with this approach.

Later, a more compact criterion for the detection of localized protrusions was proposed [24, 19]. It relies on thresholding a distance measure between the current position of virtual markers and the cell boundary at a later time point. The calculation of this distance measure, however, lacks a clear underlying definition and is computed in an ad hoc fashion for the specific data set (see supplementary material of Ref. [19]): First, each virtual marker is mapped from its current position onto the closest point on the future cell contour. As this generates a highly non-uniform distribution of target markers, with protrusive areas particularly poorly covered, the target markers on the new contour are then redistributed by two successive smoothing steps, using averaging windows of specific sizes. The time interval between the successive contours for the distance projection, as well as the smoothing parameters for redistribution of the target markers, were hand-picked by the investigator. Note, however, that this could be envisioned as one step in an iterative procedure to minimize our cost functional.

In the present work, we introduce a novel approach to define, identify and analyze localized expansions on dynamically evolving contours of amoeboid cells. expanding areas are defined via a single threshold value. In contrast to previous approaches, we chose the virtual marker dispersion as the underlying quantity, since it combines information on both, the marker displacement and the local curvature. The marker dispersion is mathematically well defined by Eq (4.26) and does not require additional empirical smoothing steps. Based on this criterion, we not only detect individual expansion events but we capture the entire shape and the complete temporal evolution of an expansion in a fully automated fashion, see Figs 4.8 and A.12 for example.

An implementation of the methodology, obtaining kymographs from regularized flows in general and detecting expansion/contraction patterns from these kymographs in particular, are fully accessible and well documented in our software Package `AmoePy`. We also provide the data of multiple cell tracks and simple artificial test cases on which the algorithm was validated. Finally, all figures and results from this article can be easily reproduced in `AmoePy`.

As outlined in the Introduction, the overall aim is a quantitative, data-driven model of amoeboid motility. The presented theoretical framework is a first step in this

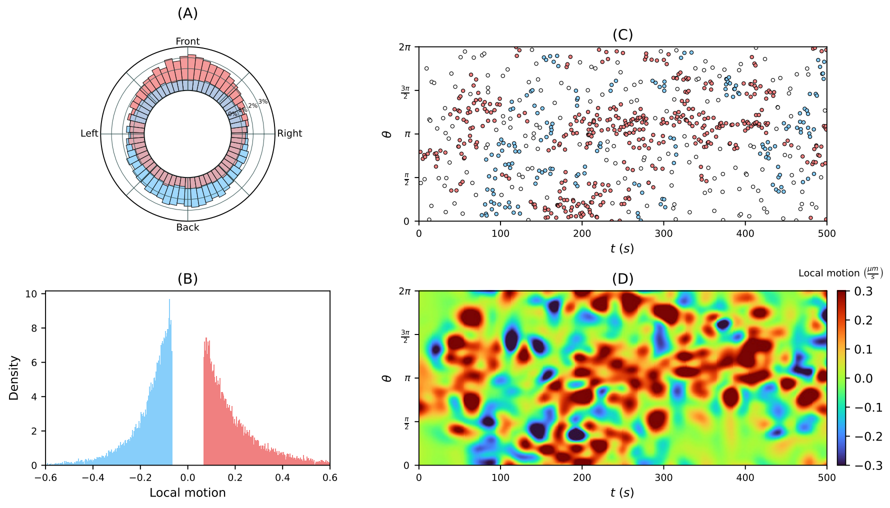


Figure 4.12. **Future outlook.** Underlying distributions are based on averaging over several cell tracks. **(A)** Circular histogram displaying the position, where expansions and contractions events appear along the cell contour. **(B)** Distribution of local motion of expansion and contractions events above a threshold $\pm 1/3$. **(C)** Simulated data obtained from self-excited Poisson point processes (so-called Hawkes processes) on the unit circle. Afterward, we obtained regions of high (red) and low (blue) intensity w.r.t. to a clustering algorithm. **(D)** Continuous kymograph obtained from a regression model (e.g. Gaussian process regression) based on sampled magnitudes at event locations shown in (C) from the distribution in (B).

direction. Because of its rigorous mathematical formulation, its efficient avoidance of mapping violations during larger shape deformations, and its moderate computational costs, it is a suitable choice for such a model. We envision that point events of high expanding activity may be used to define a point process in the space-time coordinate system. To reflect the often observed persistence in motility, so-called self-excited Poisson cluster processes or Hawkes processes may be favorable choices. The point process can serve as a skeleton for expansion activity that is 'completed' to a random realization of a kymograph, based on the statistics of a local quantity. We illustrated this idea in Fig 4.12, where we used the local motion statistics (B) to reconstruct a local motion kymograph (D). The idea is to use such realizations of kymographs to reconstruct a cell track and eventually to assimilate the time-lapse microscopy data into a mathematical model of amoeboid motility.

Modeling amoeboid cell motility

5

5.1 Currently used cell motility models

A wide variety of amoeboid motility models have been proposed tackling different key aspects of amoeboid migration such as membrane protrusion and retractions, trajectories of the cell's centroid, its polarization, and the influence of chemoattractant cues. Many proposed models are based on the concentration of interior biochemical compounds of the cell [34, 35]. In these reaction-diffusion models, motility patterns often result directly from the interplay of different processes controlling local excitation and global inhibition of the intracellular signaling and cytoskeletal dynamics[118, 119, 36, 37]. Reaction-diffusion models have been used to describe the self-organized polarization of the cell in the presence of chemoattractants [38, 39, 40, 41, 120], and the occurrence of intracellular waves and oscillations [121, 122, 123]. A large number of these approaches are based on phase-field models which are used to describe the transition between different phases such as liquid and solid states or the interior and exterior of the cell. In the latter case, one modeling approach is to define the interior and exterior of the cell as binary states with a smooth transition function. The cell membrane is then defined where the transition function reaches the exact midpoint of both states [14]. Phase-field models are often used to describe *D. discoideum* [14, 75, 40, 13, 124], but also for other cell types [125, 126, 127, 128, 129].

Other approaches are mechanical models in which different forces affect the cell motility from within and from the outside. Mechanical models differ in complexity and dimensionality to target different problems, e.g., the formation of fibroblasts (1D) [42], the influence of contraction and adhesion sites to the substratum on amoeboid cell motility (2D) [43], and the evolution of the cell surface obtained from triangulation under chemotaxis (3D) [130]. Furthermore, different physical methods and assumptions are used for the underlying model equations such as active gel physics where the gel consists of polymer filaments permeated by a solvent [131], hydrodynamics to model the internal cytoplasmic fluid [132], and the modeling of the extra-cellular matrix [133, 15]. Most importantly, mechanical models can be easily adjusted for unusual types of cell migration such as amoeboid swimmers [134, 135]. In [44], a mechanochemical model is proposed for which the underlying parameters are calibrated by using Bayesian optimization.

Finally, level set methods are used to simulate cell tracks, e.g., as part of a mechanical model of the cell cortex combined with an excitable network acting as activator/inhibitor system [35]. The excitable network is triggered by random fluctuations and can be enhanced with additional gradient stimuli and polarization modules [136, 137]. In [138], the stochastic extension of pseudopods during chemotaxis is modeled. Furthermore, stochastic differential equations have been used to model the trajectory of the cell's centroid, e.g., by a (generalized) Langevin equation [10, 30, 11, 32, 31]. In [33], the centroid trajectory was modeled by a Markov chain approach on a discrete domain obtained from hexagonal tiling. Geometric equations of evolving curves are commonly used to describe cell migration [37, 130, 139, 140]. A novel contour evolution method based on the curve-shortening flow (CSF) or, alternatively, with an optional additive function which is then called forced CSF, is presented in [141] and then applied to cell migration in [142]. A comparison of the different model approaches mentioned above can be found in [143, 144, 145].

In contrast to the above approaches, our model is designed to infer key characteristics of individual cell tracks, i.e., the intensity of protrusions and retractions during amoeboid cell motility. The model is therefore intended to be simple and intuitively comprehensible to ensure a fast and straightforward estimation of underlying model parameters and to be capable of producing a specific motility behavior for a given input. Especially the second property is sometimes hard to achieve with mechanical models which tend to have a higher complexity with a large number of model parameters and entangled subprocesses, making it difficult to draw a direct link between model parameters and a desired motility outcome [15]. By inferring the above characteristics, our model can also be used to identify differences between cells and to classify them. The second main goal of our model is to simulate a variety of quantitatively different and realistic contour dynamics producing cell tracks which can be hardly distinguished from experimental ones. Based on the model's capability to simulate such versatile contour dynamics, we deem it to be applicable to experimental cell tracks for varying degrees of motility and persistence, or even different types of locomotion.

Briefly, our model evolves two-dimensional contours representing the cell membrane and is based on three components. The first component is driven by a stochastic term generating membrane protrusions and is modeled by (1) the intensity of a self-exciting Poisson point process (so-called Hawkes process [99, 146, 100]) or, alternatively, (2) an Ornstein-Uhlenbeck like diffusion process. In this context, we show that the Hawkes process, due to its self-exciting nature, is suitable to produce a cascade of protrusion events to ensure a persistent cell migration. The second two components are mathematically well-defined geometric flows initiating contour retractions: the area-preserving curve-shortening flow (APCSF) to regularize the shape/arc length of the contour; and a further flow introduced as area adjustment flow (AAF) which expands/shrinks the cell contour with respect to a specified reference area. For more information on the APCSF, see [96, 97]. Based on the interplay of the above components, the model defines the formation of protrusions as well as retractions. It is therefore linked to other mechanistic models evolving the cell contour as an elastic object in time and space.

In the following, we demonstrate how our model can be used to simulate realistic cell tracks. Then, we analyze experimental cell tracks (*D. discoideum*) by inferring the model-based protrusion and retraction components. In this context, we compare the inferred protrusion component with commonly used biomarkers: the density of filamentous actin close to the membrane and its local motion. An implementation of the model as well as a graphical user interface to simulate cell tracks is provided in our Python-based toolbox `AmoePy` [DS3].

5.2 Three-component contour morphing model

5.2.1 General notations and underlying coordinate system

The notation and theoretical framework used in this work have been established in [DS1]. Primarily, this framework is used to (1) obtain smooth contour representations from stacks of segmented microscopy images and (2) track reference points (so-called virtual markers) between successive contours. More precisely, smooth contours and the corresponding contour curvature can be derived easily from a discrete set of segmentation points (so-called active contour or snake) by using a GPR based on *a priori* covariance structure given by some kernel. In our case, we used a Poisson kernel as underlying covariance function

$$k_r(\theta,\theta') = \frac{1-r^2}{1-2r\ \cos(\theta-\theta')+r^2}, \quad \theta,\theta' \in [0,2\pi),\ r\in[0,1). \tag{5.1}$$

Briefly, the choice of r_{cont} in $k_{r_{\text{cont}}}(\cdot,\cdot)$ affects the rigidity and stiffness of the resulting contour. Furthermore, an additional noise parameter σ_{noise} specifies the deviation between the initial segmentation points and the regression function obtained from the GPR.

First, we consider $K \in \mathbb{N}$ cell contours denoted by Γ_k with $k = 0,\ldots,K-1$. The corresponding time points are denoted by $t_k = k\cdot\delta t$ with $\delta t > 0$. The coordinates of these contours are given by the following mapping

$$\Phi_k : [0,2\pi) \to \mathbb{R}^2, \quad \theta \mapsto \Phi_k(\theta) = \left(\Phi_k^{(x)}(\theta), \Phi_k^{(y)}(\theta)\right), \tag{5.2}$$

where θ denotes the normalized arc length coordinate and $\Phi_k^{(x)}(\cdot)$ and $\Phi_k^{(y)}(\cdot)$ the x and y coordinates of the contour, respectively.

Noteworthy, connecting consecutive contours in time and space is intrinsically not well defined, i.e., there are multiple ways to do so [28, 27, 29]. In many approaches, varying constraints are introduced to track reference points/virtual markers from one contour to the next one, e.g., by using electrostatic field equations [28], level-set methods [27, 28], or mechanistic spring equations [29]. Naive contour mapping approaches include the propagation of virtual markers (VM) based on shortest paths to the next contour or by choosing paths in normal direction only. However, these approaches can change the order of neighboring virtual markers (so-called topological mapping violations).

In [DS1], we address this issue by proposing a novel regularizing family of contour flows, connecting consecutive contours while preserving desirable characteristics of the underlying mapping trajectories. Depending on a regularization parameter $\lambda_{\text{reg}} \geq 0$, this family of contour flows includes the two extreme cases: either enforcing shortest VM trajectories between contours ($\lambda_{\text{reg}} = 0$) or preserving equal distances between neighboring VM's for every time step ($\lambda_{\text{reg}} \gg 0$). For any flow between two contours Γ_k and Γ_{k+1}, a mapping

$$\phi_k : [0, 2\pi) \to [0, 2\pi), \quad \theta \mapsto \phi_k(\theta)$$

is induced, which describes the translation along the contour under the flow. By assuming

$$\partial_\theta \phi_k(\theta) > 0,$$

we ensure that ϕ_k is a one-to-one mapping, i.e., mapping violations between Γ_k and Γ_{k+1} do not occur. Now, we can define a virtual marker trajectory starting at $\theta_0 \in [0, 2\pi)$ by the iteration

$$\theta_{k+1} = \phi_k(\theta_k). \tag{5.3}$$

In the following, we use a Lagrangian reference frame to describe geometric flows and the resulting evolution of virtual markers on the contour. This Lagrangian reference frame is denoted by χ_k and recursively defined by:

$$\chi_{k+1}(\theta_0) = \phi_k(\chi_k(\theta_0)), \quad \chi_0(\theta_0) = \theta_0.$$

For the limit of infinitely dense contours, we introduce the following notations

$$\begin{aligned} \Phi : [0, T] \times [0, 2\pi) &\to \mathbb{R}^2 \\ (t, \theta) &\mapsto \Phi(t, \theta) \end{aligned} \quad \text{and} \quad \begin{aligned} \chi : [0, T] \times [0, 2\pi) &\to [0, 2\pi) \\ (t, \theta) &\mapsto \chi(t, \theta). \end{aligned}$$

Given a VM $p_0 = \Phi_0(\tilde{\theta})$ on the first contour Γ_0 with arc length coordinate $\tilde{\theta} \in [0, 2\pi)$, we can now track this VM in time and space which corresponds to the function $t \mapsto \Phi(t, \tilde{\theta})$.

Finally, we introduce a virtual marker distance ratio defined as:

$$\text{VMDR}_{t,\theta} = \partial_\theta \chi(t, \theta). \tag{5.4}$$

For the discrete case of $N \in \mathbb{N}$ virtual markers $\theta_{k,0}, \ldots, \theta_{k,N-1}$ on the contour Γ_k, this ratio can be rewritten as:

$$\text{VMDR}_{k,i} = \frac{|\theta_{k,i+1} - \theta_{k,i}|}{2\pi/N}, \tag{5.5}$$

measuring the distance between neighboring virtual markers divided by the distance of equidistantly spaced VM's.

Contour propagation vs. contour mapping. We need to distinguish two different dynamics, which are both compatible with a given sequence of contours: (1) the contour propagation based on a model function $f : \mathbb{R}^+ \times \mathcal{S}^1 \to \mathbb{R}$ describing the normal velocity and (2) the "material" contour mapping under which each virtual marker is transported. This results in the following two-step algorithm: the dynamic of the contour itself is obtained by propagating for short time periods an initially equidistant set of contour points with respect to the normal vector field

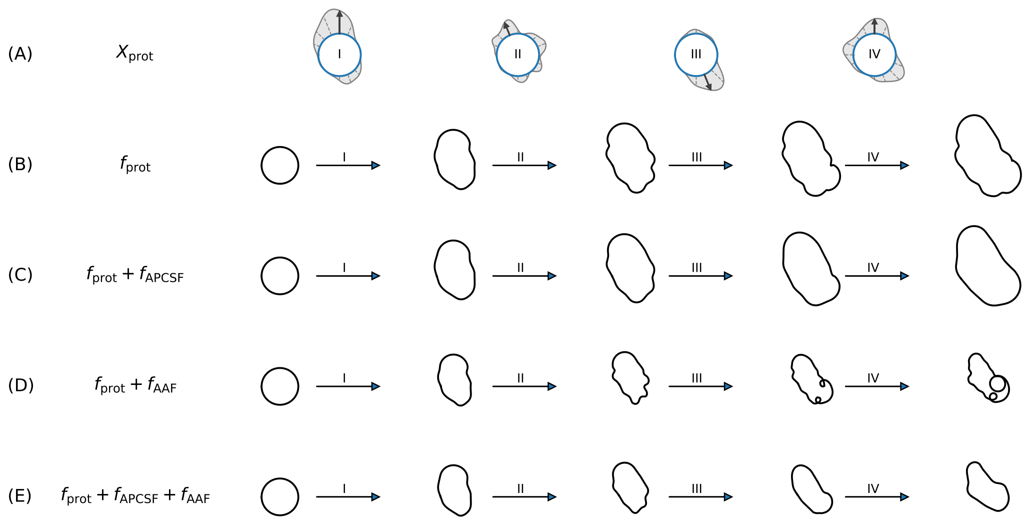
$$\left\langle \frac{\partial \Phi(t,\theta)}{\partial t}, \vec{n}(t,\theta) \right\rangle = f(t,\theta), \tag{5.6}$$

at time $t \in [0,T]$ and normalized arc length coordinate $\theta \in [0,2\pi)$. A propagation in tangential direction affects the position of these contour points but does not affect the shape of the contour. Hence, the only information we have from the contour dynamics alone is the normal component of the actual material flow as expressed in Eq (5.6). However, such normal flow leads to thinning and clustering effects of the transported points over time. For this reason, we use a second kind of dynamics, namely regularizing flows described as in [DS1] to ensure an evenly-spaced distribution of virtual markers. This regularized flow is used to transport any dynamically relevant quantity over the dynamically evolving contours. Furthermore, the flow lines give rise to a coordinate system necessary to draw graphical representations (so-called kymographs) of each model component. In contrast to the contour propagation, the virtual markers under the regularizing flow are propagated also in tangential direction. In Section B.3, we present a detailed description regarding the implementation of the model. In Fig B.3, we illustrate the two different kinds of marker trajectories: (1) the contour propagation (green dashed lines) and (2) the contour mapping (blue dashed lines) under which the stochastic protrusion process $X_{\text{prot}}(t,\theta)$ is transported and the underlying coordinate system is based on.

5.2.2 Regularizing shape and size of contours via geometric flows

Our model to evolve two-dimensional cell contours is based on three components: a protrusion term based on a stochastic process accounting for membrane protrusions and two geometric flows accounting for membrane retractions by regularizing the shape and area of the contour. Due to the separation into one protrusion component and two retraction components, the model provides an independent handling of these two key features of amoeboid cell motility.

The first geometric flow is defined by the area-preserving curve-shortening flow (APCSF) and denoted by f_{APCSF}. Briefly, the APCSF evolves a two-dimensional contour to a circle while preserving the area enclosed by the initial contour. It therefore minimizes the arc length of the contour without affecting the contour area. In the absence of other influences, the evolution of every contour to a circle is energetically favorable to reduce the surface tension of the membrane. For example, this behavior can be observed during cell death or by treating cells with Latrunculin to dissolve the actin cytoskeleton [147]. The second geometric flow which we call "area adjustment flow" (AAF) is denoted by f_{AAF}. The AAF shrinks/expands the contour towards a predefined reference area. Since the underlying contour data



(A)
X_{prot}
I
II
III
IV
(B)
f_{prot}
(C)
$f_{\text{prot}} + f_{\text{APCSF}}$
(D)
$f_{\text{prot}} + f_{\text{AAF}}$
(E)
$f_{\text{prot}} + f_{\text{APCSF}} + f_{\text{AAF}}$

Figure 5.1. Rationale why all three model components are necessary. (A) Realizations of stochastic process X_{prot} driving the protrusion component f_{prot}. **(B)** Simulated contour dynamics only based on protrusion component leading to significant contour growth. **(C)** By combining the protrusion component with the APCSF, we obtain smoother and similarly sized contours. **(D)** On the contrary, combining the protrusion component with the AAF only results in highly curved contours with possible self-intersections. **(E)** A combination of all three components is necessary to obtain stable contour dynamics over time.

relies on two-dimensional cross sections of three-dimensional cells, the resulting contour area can change significantly. Therefore, a certain change of the contour area is desirable and possible in our model. By using both flows, we achieve a regularizing effect on the arc length (APCSF) and the area (AAF) necessary to counteract the forward movement initialized by the protrusion component f_{prot}. This component is based on a stochastic process, e.g., a Hawkes intensity process or an Ornstein-Uhlenbeck process, and should be strictly positive since it describes the formation of protrusions only.

In Fig 5.1, simulated contour dynamics are shown based on: the protrusion component f_{prot} only, f_{prot} combined with f_{APCSF}, f_{prot} combined with f_{AAF}, and a combination of all three components. In the first row, four different samples drawn from the same stochastic process X_{prot} are displayed. The first two cases (panels (B) and C)) lead to significantly growing contours since the area adjustment is absent. In comparison, the contour dynamics in panel (C) are much smoother due to the regularizing effect of the APCSF. In panel (D), dynamics with highly curved but similarly sized contours are produced. However, self-intersections can easily occur without the APCSF. Therefore, a combination of all three components, as in our model, is necessary (last row).

Area-preserving curve-shortening flow. To regularize the contour arc length in our model, we used the APCSF:

$$\left\langle \frac{\partial\Phi(t,\theta)}{\partial t}, \vec{n}(t,\theta) \right\rangle = -\left(\kappa(t,\theta) - \frac{2\pi}{L(\Phi(t,\cdot))} \right), \tag{5.7}$$

where $\kappa(t,\theta)$ denotes the curvature and $\vec{n}(t,\theta)$ the outward-pointing normal vector at time $t \in \mathbb{R}^+$ for a virtual marker with arc length coordinate $\theta \in [0, 2\pi)$ on the first contour. Here, $L(\Phi(t,\cdot))$ denotes the total arc length and is defined by the following functional:

$$L(t) = L\left(\Phi(t,\cdot)\right) = \int_0^{2\pi} \left\| \frac{\partial\Phi(t,\theta)}{\partial\theta} \right\|_2 d\theta.$$

The APCSF is defined as the gradient flow of this functional under the area-preserving constraint $A(t) := A(\Phi(t,\cdot)) = A(0)$ for all $t > 0$, where the contour area is defined by:

$$A(t) = A(\Phi(t,\cdot)) := \int_0^{2\pi} \Phi^{(x)} \frac{\partial\Phi^{(y)}}{\partial\theta}\, d\theta = \int_0^{2\pi} \Phi^{(y)} \frac{\partial\Phi^{(x)}}{\partial\theta}\, d\theta.$$

As a consequence, the APCSF evolves every contour to a circle of the same area, minimizing the contour arc length to $L(t) \xrightarrow{t\to\infty} 2\sqrt{\pi A(0)}$ while maintaining its area.

Area adjustment flow. While the APCSF has no effect on the contour area, the protrusion component would expand the cell contour most of the time and therefore also its area. The area adjustment flow counteracts this expansion. It is defined as

$$\left\langle \frac{\partial\Phi(t,\theta)}{\partial t}, \vec{n}(t,\theta) \right\rangle = -\frac{A(t) - A_{\text{ref}}}{A_{\text{ref}} \cdot L(t)} \left\langle \Phi(t,\theta) - \Phi_{\text{CM}}(t), \vec{n}(t,\theta) \right\rangle \tag{5.8}$$

with reference area $A_{\text{ref}} \in \mathbb{R}^+$ and center of mass trajectory $\Phi_{\text{CM}} : \mathbb{R}^+ \to \mathbb{R}^2$ defined by

$$\Phi_{\text{CM}}(t) = \frac{1}{2\pi} \int_0^{2\pi} \Phi(t,\theta) d\theta.$$

For a choice of the reference area A_{ref}, we take the 1st percentile of the entire area time series of an experimental cell tracks. Other choices are possible.

It is easy to see that an area adjustment is achieved by this flow. Furthermore, the AAF affects the contour in normal and tangential direction and is shape-preserving if used *without* the other two components. In Section 3.2.2, we introduced another area regularizing flow which is based on the gradient flow of the area functional. This flow minimizes the contour area much more rapidly by affecting the contour in normal direction only. However, this flow is not shape-preserving.

5.2.3 Hawkes intensity process as protrusion component

Statistical analyses of different sequences of cell contours have shown that a protrusion event increases the probability of nearby follow-up protrusions [6]. We therefore modeled the protrusion component f_{prot} in our model by the intensity of a self-exciting Poisson point process, a so-called Hawkes process. Due to the self-exciting nature of the Hawkes process, a cascade of protrusion events can be generated which result in substantial and persistent contour changes. In this way, we obtain contour dynamics with a significantly moving center of mass instead of a fluctuating membrane only.

The intensity $\lambda(t, \theta)$ of a spatio-temporal Hawkes process is defined by

$$\lambda(t, \theta) = \mu(\theta) + \sum_{i:t_i<t} g\left(t - t_i, \theta - \theta_i\right), \tag{5.9}$$

with event times $\{t_1, t_2, \dots\}$, background intensity function $\mu : [0, 2\pi) \to \mathbb{R}^+$ and kernel function $g : [0, T] \times [0, 2\pi) \to \mathbb{R}^+$, characterizing the positive influence of past events (ancestors) on the emergence of future events (descendants).

The background intensity is defined in terms of the normalized Poisson kernel $\tilde{k}_{r_{\text{pol}}}$:

$$\tilde{k}_r(\theta, \theta') = \frac{k_r(\theta, \theta')}{\sqrt{k_r(\theta, \theta) k_r(\theta', \theta')}} = \frac{(1-r)^2}{1 - 2r\ \cos(\theta - \theta') + r^2}, \tag{5.10}$$

with $\theta, \theta' \in [0, 2\pi)$ and $r \in [0, 1)$. The normalized Poisson kernel holds the following properties: $\left(\frac{1-r}{1+r}\right)^2 \le \tilde{k}_r(\theta, \theta') < 1$ for all $\theta \neq \theta'$ and $\tilde{k}_r(\theta, \theta') = 1$ if and only if $\theta = \theta'$. In Fig B.4, the Poisson kernel function from Eq (5.1) and its normalized version from Eq (5.10) are shown for different parameters $r \in [0, 1)$. As background intensity function, we now define

$$\mu(\theta) = \lambda_0 \frac{\tilde{k}_{r_{\text{pol}}}(\theta, \pi)}{\int_0^{2\pi} \tilde{k}_{r_{\text{pol}}}(\theta, \pi)\, d\theta}, \tag{5.11}$$

with background rate $\lambda_0 > 0$ and $\tilde{k}_{r_{\text{pol}}}$ as in Eq (5.10) with $0 \le r_{\text{pol}} < 1$. For the non-polarized case $r_{\text{pol}} = 0$, the background intensity simplifies to $\mu = \frac{\lambda_0}{2\pi}$. For $r_{\text{pol}} > 0$, polarization takes place with a local maximum at $\theta = \pi$, and local minima at $\theta = 0$ and $\theta \to 2\pi$.

We used a product kernel function $g(t, \theta) = g_1(t) \cdot g_2(\theta)$ with temporal component $g_1(t)$ and spatial component $g_2(\theta)$. The temporal kernel function is given by

$$g_1(t) = \alpha\beta\, t\, e^{-\beta t},$$

with arrival intensity $\alpha > 0$ and exponential decay rate $\beta > 0$. As spatial kernel, we used the von Mises distribution,

$$g_2(\theta) = \frac{e^{\kappa_M \cos(\theta)}}{2\pi I_0(\kappa_M)},$$

with $\kappa_M > 0$ as concentration parameter and $I_0(\kappa_M)$ denoting the modified Bessel function of order 0.

For a realization of the Hawkes process, the protrusion process $X_{\text{prot}} : \mathbb{R}^+ \times \mathcal{S}^1 \to \mathbb{R}$ is defined as:

$$X_{\text{prot}}(t,\theta) = \frac{c_s}{\text{VMDR}(t,\theta)} \sum_{i:t_i<t} \tilde{g}\left(t - t_i, \theta - \theta_i\right), \tag{5.12}$$

with any spatio-temporal kernel function $\tilde{g} : [0,T] \times [0,2\pi) \to \mathbb{R}^+$, time scaling factor $c_s > 0$, and VMDR as in Eq (5.4) accounting for local contour/arc length changes. For the sake of simplicity, we choose $c_s = 1s$ and the same kernel function as above, i.e., $\tilde{g}(t,\theta) \equiv g(t,\theta)$. Hence, we can rewrite the protrusion process X_{prot} in terms of the Hawkes intensity process $\lambda(t,\theta)$ from Eq (5.9):

$$X_{\text{prot}}(t,\theta) = \frac{c_s}{\text{VMDR}(t,\theta)} \left(\lambda(t,\theta) - \mu(\theta)\right), \tag{5.13}$$

to highlight that X_{prot} directly resembles the intensity of the Hawkes process and not the underlying realization of point events or the Hawkes process itself.

Based on the underlying choice of parameters, different motility characteristics can be adjusted with our model:

- the general movement speed by w_{prot},
- the number of protrusions by λ_0 and α,
- the duration of protrusions by β,
- the size of protrusions (many small protrusions vs. a single large protrusion) by κ_M,
- membrane fluctuation vs. creation of pseudopods with a substantial movement of the center of mass by α,
- non-polarized vs. polarized contour dynamics by r_{pol}.

In Fig B.5, the kernel functions $g_1(t)$ and $g_2(\theta)$ are illustrated for varying parameters α, β, and κ_M as well as the Hawkes intensity $\lambda(t,\theta)$ for a fixed set of parameters. In Section B.5, we illustrate an alternative approach by modeling the protrusion component with an Ornstein-Uhlenbeck type of diffusion process.

5.2.4 Three-component contour dynamics model

In this section, we formulate our contour dynamics model based on the three components: a protrusion component based on a stochastic process X_{prot} from Eq (5.13), the APCSF from Eq (5.7), and the AAF from Eq (5.8). In our model, the following parameters are required:

- weight parameters $w_{\text{prot}}, w_{\text{APCSF}}, w_{\text{AAF}} > 0$
- a reference area $A_{\text{ref}} > 0$,
- additional parameters regarding the stochastic process X_{prot}.

Furthermore, the computation of the following geometric quantities is necessary:

- contour area $A(t) \in \mathbb{R}^+$ and arc length $L(t) \in \mathbb{R}^+$,
- contour curvature $\kappa(t,\theta) \in \mathbb{R}$,

- center of mass trajectory $\Phi_{\text{CM}}(t) \in \mathbb{R}^2$,
- outward-pointing unit normal vectors $\vec{n}(t,\theta) \in \mathbb{R}^2$.

For a virtual marker $\Phi(0,\theta)$ with initial arc length coordinate $\theta \in [0, 2\pi)$, the normal component of its trajectory $t \mapsto \Phi(t,\theta)$ with $t \in [0,T]$ is given by

$$\left\langle \frac{\partial \Phi(t,\theta)}{\partial t}, \vec{n}(t,\theta) \right\rangle = f(t,\theta), \tag{5.14}$$

where $f : \mathbb{R}^+ \times \mathcal{S}^1 \to \mathbb{R}$ is defined as

$$f = f_{\text{prot}} + f_{\text{APCSF}} + f_{\text{AAF}}, \tag{5.15}$$

with the following three components:

$$\begin{aligned}
I: \qquad & f_{\text{prot}}(t,\theta) &&= w_{\text{prot}} \frac{X_{\text{prot}}(t,\theta)}{L(t)}, \\
II: \qquad & f_{\text{APCSF}}(t,\theta) &&= -w_{\text{APCSF}} \left(\kappa(t,\theta) - \frac{2\pi}{L(t)} \right), \\
III: \qquad & f_{\text{AAF}}(t,\theta) &&= -w_{\text{AAF}} \frac{A(t) - A_{\text{ref}}}{A_{\text{ref}} \cdot L(t)} \left\langle \Phi(t,\theta) - \Phi_{\text{CM}}(t), \vec{n}(t,\theta) \right\rangle
\end{aligned} \tag{5.16}$$

In Fig 5.2, these three components are displayed. In panel (A), the protrusion component is displayed with a generated sample of the underlying stochastic process X_{prot} in the top right square. X_{prot} is defined on a unit circle and is mapped onto the contour with respect to the contour arc length and a reference point $\theta = 0$ (blue dot). In panel (B), the APCSF is shown with its final state (dashed grey contour): a circle with the same area as the initial contour. Since the APCSF is mainly defined by the contour curvature, the contour shrinks faster for regions with large positive (convex) curvature and expands faster for regions with large negative (concave) curvature. In panel (C), it can be observed that the area adjustment is always acting in the direction towards the center of mass. The final state is again displayed as a dashed gray contour.

Finally, in panels (D-F), we display test scenarios of simulated contour dynamics each based on a single component only. In panel (D), we observe a steadily growing contour since the f_{prot} is the only active component. Under the APCSF (panel (E)) the contour evolves to a circle by expanding concave parts and shrinking convex parts of the contour. In the case of AAF being the only active component, we observe a shape-preserving shrinkage of the contour. In this case, the reference area ($A_{\text{ref}} = 60\mu m^2$) was set to be smaller than the initial contour area. Alternatively, by choosing a larger reference area, a shape-preserving growth of the contour would occur.

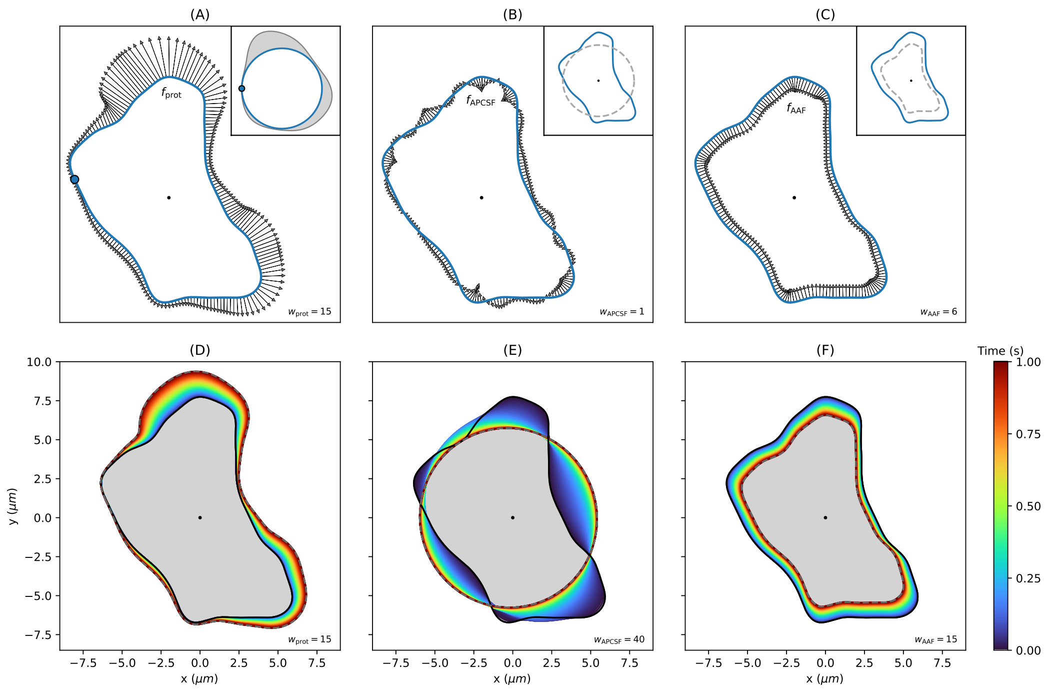


Figure 5.2. Comparison of the three model components: f_{prot}, f_{APCSF}, and, f_{AAF}. (A) Protrusion component driven by a stochastic process (generated sample shown in top right corner). The reference point $\theta = 0$ is highlighted as blue dot. **(B, C)** APCSF and AAF with final states displayed in corresponding top right corners. **(D-F)** Example contour propagated individually by only one component for a time span of $1s$. Regarding the AAF, the underlying reference area is given by $A_{\text{ref}} = 60\mu m^2$.

5.2.5 Inference of model components and parameter estimation

Besides simulating contour dynamics, our model is used to infer the different model components $f_{\text{prot}}, f_{\text{APCSF}}, f_{\text{AAF}}$ of experimental cell tracks to analyze and characterize cell tracks on the individual level. Importantly, we want to quantify the intensity of protrusions and retractions separately from each other. Usually, this is done by computing the local motion which, however, comprises the characteristics of the entire contour dynamics: protrusions, retractions, as well as minor membrane fluctuations. Finally, by inferring the underlying model components and estimating the corresponding component weights, our aim is to classify cells based on different motility types.

Since the APCSF and AAF from Eq (5.16) are based on the contour curvature, arc length, and area; f_{APCSF} and f_{AAF} can be computed separately for each time step solely based on the current contour. For this reason, one can explicitly determine the only remaining component f_{prot} to propagate one contour to the next one. This approach enables us to replicate the experimental cell track with our model for an estimated set of model parameters. This set includes the reference area A_{ref} and three model component weights w_{prot}, w_{APCSF}, and w_{AAF}.

While the APCSF does not affect the contour area, the (positive) protrusion component increases the contour area. Therefore, the reference area A_{ref} should be chosen relatively small to achieve a counteracting shrinkage effect of the AAF. In our case, we have chosen the 1st percentile of the entire area time series of the cell track. Next, we estimate f_{APCSF} and f_{AAF} and their corresponding weights, w_{APCSF} and w_{AAF}, by making use of the local motion kymograph of the underlying cell track. More precisely, we determine w_{APCSF} and w_{AAF} such that negative regions (i.e. retractions) of the local motion are optimally captured by the above components. Initially, the remaining component f_{prot} is estimated by deducting f_{APCSF} and f_{AAF} from the local motion kymograph. Afterwards, we choose w_{prot} such that the sample variance of the underlying protrusion process X_{prot} is standardized with $\text{Var}(X_{\text{prot}}) = 1$. In the next step, we further tune the inferred protrusion component f_{prot} by minimizing the distances of virtual markers propagated with respect to f_{prot} and the "receiving" contour at the next time step. In this optimization step, we use the built-in least-squares method from the Python package `SciPy`.

Finally, we evaluate the goodness of fit of the inferred protrusion component. Since f_{APCSF} and f_{AAF} are computed in advance, the remaining protrusion component corrects any missing contour dynamics to propagate one contour to the next one. For this reason, the inferred protrusion component can be negative if a contour retraction is stronger than the APCSF and the AAF have predicted. Therefore, a good fit is given if the estimated protrusion component is (mostly) positive, i.e., the contour retractions are successfully captured by the other two components.

5.2.6 Hawkes process based simulations of amoeboid cell motility

In this section, we show that the Hawkes process is suitable to simulate ameboid cell motility. With the proposed model a variety of qualitatively different contour dynamics were generated.

The protrusion component in this section is based on a Hawkes process defined as in Eq (5.13). The underlying model weights were set to $w_{\text{prot}} = 7\mu m^2/s$, $w_{\text{APCSF}} = 0.1\mu m^2/s$, and $w_{\text{AAF}} = 1\mu m/s$. As reference area we have chosen $A_{\text{ref}} = 80\mu m^2$. First, we simulated contour dynamics based on a non-polarized test scenario realized by choosing $r_{\text{pol}} = 0$. Then, we have chosen a polarized test scenario by choosing $r_{\text{pol}} = 0.5$. A summary of all parameter values is displayed in Table 5.1. As initial contour we have chosen a circle with an area equal to A_{ref}.

Fig 5.3 shows exemplary non-polarized contour dynamics, which are driven by a Hawkes process. We observed that the Hawkes process can enforce a substantial change of the center of mass trajectory as displayed in panels (A) and (B). The self-excitation can be also observed in the kymograph of the protrusion component f_{prot} (panel (D)). In this kymograph, point events realized from the Hawkes process are depicted as circles and show clusters as expected from the self-excitation property. The corresponding Hawkes intensity was then used to define the protrusion process

Table 5.1. **Choice of parameters and meaning.**

Parameter	Value	Unit	Meaning
Contour parametrization			
r_{cont}	0.6	–	GPR smoothing
σ_{noise}	0.05	–	GPR noise
λ_{reg}	10	$\frac{\mu m^2}{s^2}$	Flow regularization
A_{ref}	80	μm^2	Reference area
Hawkes process			
λ_0	1	s^{-1}	Background intensity
α	0.4	s^{-1}	Arrival intensity
β	0.5	s^{-1}	Exponential decay rate
κ_M	100	–	Spatial concentration
r_{pol}	0 and 0.5	–	Polarization
Model weights			
w_{prot}	15	$\frac{\mu m^2}{s}$	Protrusion weight
w_{APCSF}	0.1	$\frac{\mu m^2}{s}$	APCSF weight
w_{AAF}	1	$\frac{\mu m}{s}$	AAF weight

List of parameters for simulated contour dynamics based on a Hawkes process.

in our contour dynamics model as in Eq (5.13). The same red patterns of the f_{prot} kymograph can be also found in the final local motion kymograph (panel (C)). The retractions of the contour dynamics are mainly defined by the area adjustment f_{AAF} (panel (E)). By comparing this kymograph with the area plot in panel (G), we see that dark blue patterns, indicating strong retractions, occur when the contour area $A(t)$ is much larger than the reference area A_{ref}.

Due to the definition of the APCSF component f_{APCSF}, the curvature at each part of the contour can be inferred from the corresponding kymograph (panel (F)). Since the contour dynamics start from a perfect circle, possessing a constant and relatively small curvature, the kymograph starts with values close to zero. Later on, blue horizontal stripes indicate the position of protrusions and the rear of the cell, which possess a large positive curvature and are retracted therefore by the APCSF. In contrast, concave regions of the cell contour correspond to red horizontal stripes since they are enforced to expand under the APCSF. Finally, the influence of the APCSF, minimizing the contour arc length, is shown in the plot in panel (H).

Fig 5.4 shows an artificial cell track based on a polarized Hawkes process. In contrast to the cell track of Fig 5.3, the contour dynamics a characterized by a higher motility and a stronger persistence. Interestingly, we observed the zigzag pattern well known from experimental amoeboid cell tracks [6]. The self-exciting behavior of the Hawkes process initialized with a spatially unimodal background intensity (see Fig B.4) is sufficient to reproduce the zigzag movement. For this case, the clustering

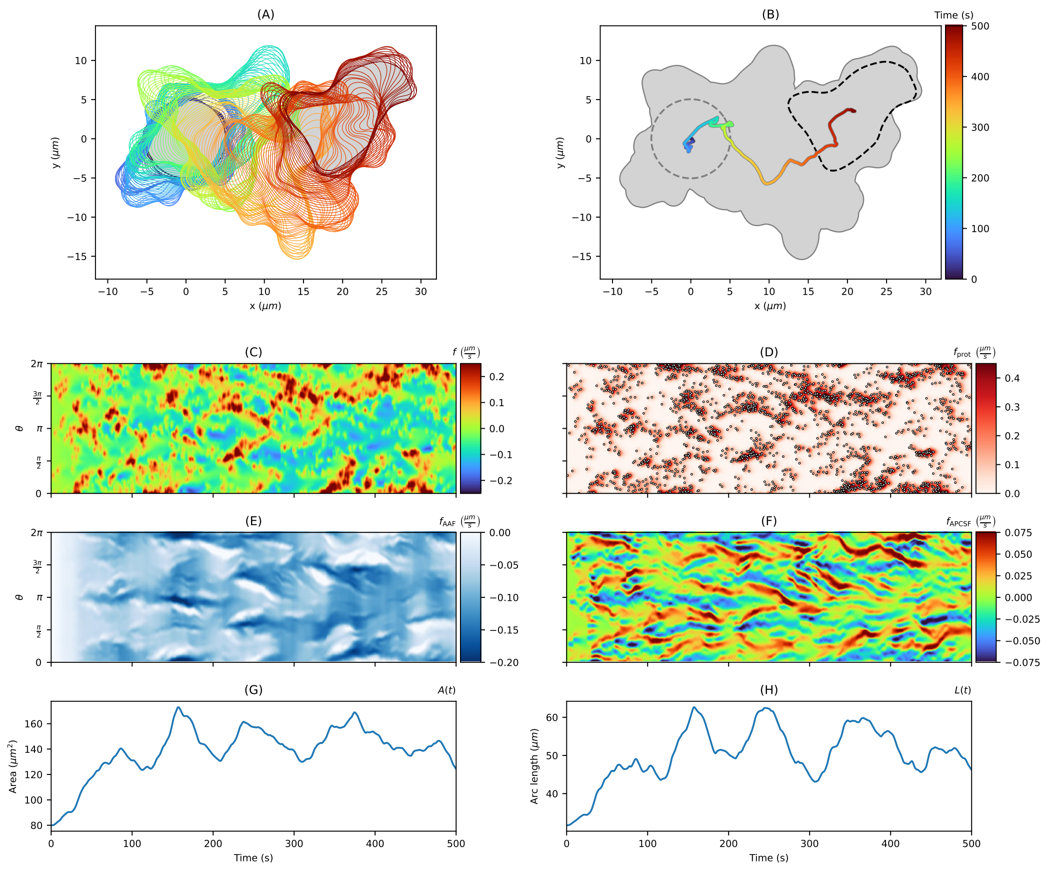


Figure 5.3. Artificial non-polarized cell track with $T = 500s$ based on a Hawkes process. (A, B) Contour dynamics (left, colored contours), the center of mass trajectory (right, colored line), and the trace of the entire cell track(right, gray area). **(C, D)** In the second row, kymographs of the local motion f and its protrusion component f_{prot} are displayed. Point events realized by the underlying Hawkes process are depicted as circles in the protrusion kymograph. **(E, F)** In the third row, kymographs of the other two components f_{AAF} and f_{APCSF} are shown. Finally, in panels **(G, H)**, the evolution of the contour area as well as the contour arc length are presented.

of point events (panel (D)) is even more pronounced than for the non-polarized case. From the local motion kymograph, we infer that the protrusions (red regions) occur mainly at $\theta = \pi$ defining the front of the cell. On the other side, retractions (blue regions) occur near $\theta = 0$ and $\theta = 2\pi$. In contrast to the non-polarized scenario, the curvature lines in the kymograph representing f_{APCSF} are less horizontal. Instead, the characteristic diagonal stripes stand for protrusions created at the front of the cell which are then moved along both sides of the contour until they reach the rear of the cell. This has also been observed experimentally in [19].

In addition to the Hawkes process, we also modeled the creation of protrusions by a simple Poisson point process. We generated cell tracks for each of the two scenarios described above where the protrusion component is only defined by the first generation of point events without further offspring as for the Hawkes process.

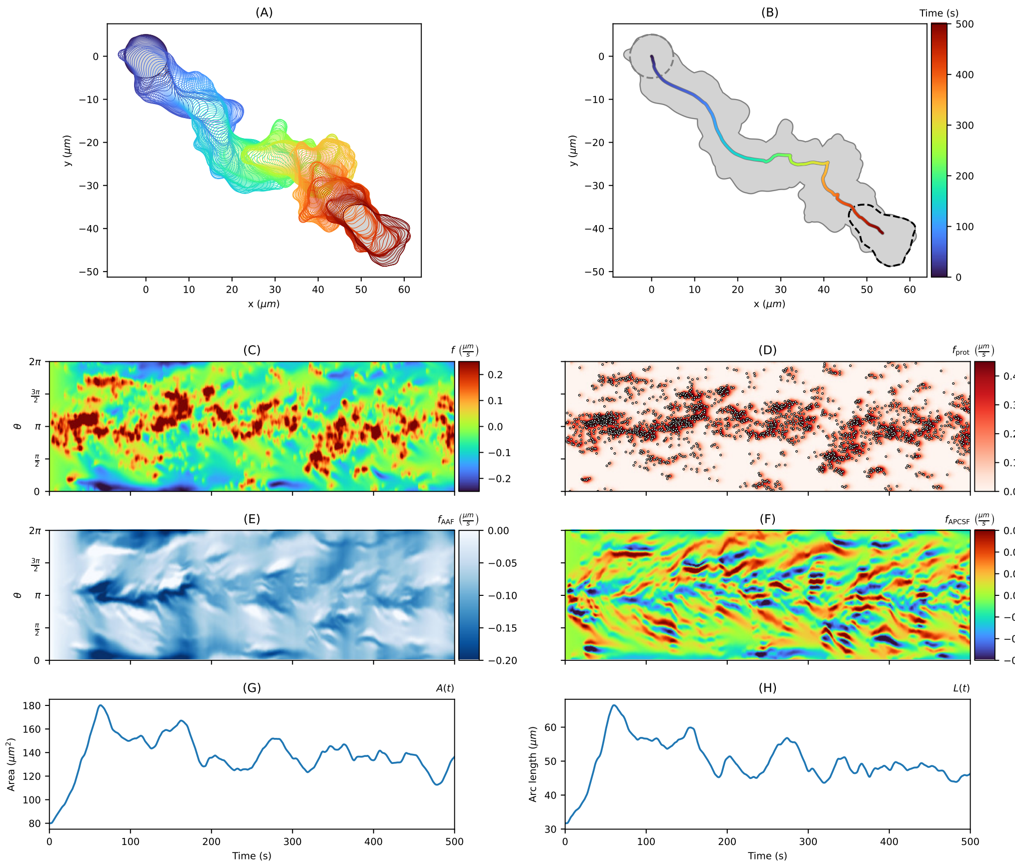


Figure 5.4. Artificial polarized cell track with $T = 500s$ based on a Hawkes process. The contour dynamics, the center of mass trajectory, the contour trace, the kymographs of f, f_{prot}, f_{APCSF}, and f_{AAF} as well as the contour area and arc length are shown in the same order as in Fig 5.3.

To achieve the same number of point events, we increased the background intensity from $\lambda_0 = 1s^{-1}$ to $\lambda_0 = 5s^{-1}$.

In Figs B.6 and B.7, five cell tracks driven by such a standard (i.e. not self-exciting) Poisson process are displayed, respectively. In comparison to the cell tracks generated by the Hawkes process, we observe a more equal distribution of point events with less significant event clusters. Therefore, for the non-polarized scenario, membrane fluctuations without large displacements of the center of mass were observed. For the polarized scenario, the cell moves persistently in one direction without having major turns, zigzag movements, or additional *de novo* pseudopodia. Hence, the Hawkes process was better suited to model amoeboid cell motility than a standard Poisson point process.

Animations of the contour dynamics and the corresponding kymographs from Fig 5.3 and Fig 5.4 are shown in Video B.1 and Video B.2, respectively. In these videos, each point event generated by the Hawkes process is illustrated as a circle in the protrusion component kymograph (second row) and the animated contour dynamics (bottom left). The effect of the self-excitation of the Hawkes process can be clearly seen in form of cascades of point events determining the direction and the movement of the cell.

We furthermore examined a second alternative to the Hawkes process, an Ornstein-Uhlenbeck type process, see Section B.5 for more details. However, the resulting simulations are less satisfactory due to a higher number of protrusive features that are not observed in experiments and additional artifacts such as a pulsating membrane and a swimming type of locomotion. Since a self-exciting property is missing in this approach, it is much more difficult to generate cascades of multiple protrusions and subsequent reorientation phases of the contour dynamics. Contour dynamics and the corresponding model components obtained from this approach are displayed in Section B.5 with animations shown in Video B.3.

How to choose the regularization parameter λ_{reg}? To ensure stable contour dynamics within our model, we used regularized flows defined as in [DS1]. Based on a regularization parameter $\lambda_{\text{reg}} \geq 0$, we can enforce a more even distribution of virtual markers for every time step/contour. This way, thinning/clustering effects of virtual markers over time can be avoided, see Section B.3 and FigB.3 for more details.

In this section, we investigated the influence of this regularization on the overall contour dynamics and studied under which circumstances clustering/thinning effects of virtual markers as well as other artifacts can occur. Moreover, we challenged our model by varying the temporal resolution. In this case, the underlying contour mapping is also affected since the number of contours is increased/reduced.

In Fig 5.5, we present simulations of non-polarized cell tracks (panel (B)) as well as polarized cell tracks (panel (C)) generated with the parameter choices in Table 5.1 but for varying regularization parameter $\lambda_{\text{reg}} \in \{0.01, 0.1, 10, 1000\}$. The trace of each cell track (gray area) and the corresponding centroid trajectories (colored lines) are displayed. For the non-polarized case (panel (B)), the overall contour dynamics were substantially altered in cases of weak ($\lambda_{\text{reg}} = 0.01$) or strong regularization ($\lambda_{\text{reg}} = 1000$). For a medium regularization ($\lambda_{\text{reg}} = 0.1, 10$), the model produced contour dynamics similar to each other with almost identical contours at the end of each track. Analogous observations were made for the polarized test cases in panel (C). By decreasing λ_{reg}, the overall motility is also reduced. This can be understood from Eq (5.13): a low regularization results in thinning effects of VM's at the leading edge which increases the virtual marker distance ratio (VMDR) and therefore decreases the protrusion process. In panel (C), we noticed differences in the main direction of the cell track due to different contour mappings at the beginning of each track. However, the overall contour dynamics was not affected by increasing λ_{reg}.

In panels (D) and (E), histograms are displayed showing the effect of each regularization scheme on the distribution of virtual markers for non-polarized (left) and polarized (right) contour dynamics. Each histogram displays the relative frequencies with respect to the virtual marker distance ratio from Eq (5.5) for all contours of a simulated cell track. In case of a weak regularization, the thinning and clustering effect are reflected by a higher variance of the VMDR with distances up to 2 times larger than in the equidistant case. Especially in panel (E) for $\lambda_{\text{reg}} = 0.01$ (light blue histogram), a major peak at $\text{VMDR} \approx 0.5$ indicates a strong clustering of virtual markers at the rear of the cell. In case of a strong regularization, an almost even distribution of virtual markers was achieved for every time step/contour which is reflected by VM distance ratios close to 1.

In panels (F) and (G), kymographs of the APCSF component corresponding to the non-polarized (left) and polarized (right) tracks are displayed. For a weak regularization ($\lambda_{\text{reg}} = 0.01$), prominent thinning and clustering effects of virtual markers were observed, see dashed box in panel (G). In case of a strong regularization ($\lambda_{\text{reg}} = 1000$), an equidistant distribution of virtual markers is enforced at each time step. As a consequence, an emerging protrusion has a direct effect on all virtual markers along the cell contour leading to a cyclic shift of all markers. These shifts were observed as rotating elements of the cell contour indicated by sharp diagonals in the APCSF kymograph, see dashed boxes in panel (F). For medium regularization choices ($\lambda_{\text{reg}} = 0.1, 10$), the above artifacts (clustering effects, rotating elements) were not observed.

A summary of our findings is presented in Fig 5.5(A), where the influence of λ_{reg} on the distribution of virtual markers is shown. In this context, we computed the standard deviation σ_{VMDR} of the virtual marker distance ratio for varying λ_{reg}. For a weak regularization $\lambda_{\text{reg}} = 0.01$, a loose connection of neighboring virtual markers is observed which leads to thinning and clustering effects. If the regularization is chosen too strong ($\lambda_{\text{reg}} = 10^2, 10^3$), other artifacts such as rotating elements of the contour might occur. For this reason, we recommend a medium regularization: $0.1 \leq \lambda_{\text{reg}} \leq 10$. In our simulations, we have chosen the upper limit $\lambda_{\text{reg}} = 10$ to obtain a more equidistant distribution of virtual markers which ensures stable contour dynamics even for a longer time period $T > 500s$ while avoiding rotating artifacts of the contour.

For more details, see Fig B.9, where the above non-polarized cell tracks are shown for a wider selection of varying regularization parameters $\lambda_{\text{reg}} \in \{10^{-2}, 10^{-1}, \dots, 10^3\}$ and with all corresponding model components. Again, for the intermediate cases $\lambda_{\text{reg}} \in [0.1, 10]$, we noticed only very few differences in all shown kymographs. However, because of small changes of the contour mappings at each time step, the overall direction of the cell track can vary over time.

The polarized cell tracks under varying regularization schemes are shown in Fig B.10. As mentioned above, clustering and thinning effects of virtual markers were prominent for $\lambda_{\text{reg}} = 0.01$. For the other choices of λ_{reg}, we observed that all kymographs are barely affected at all, resulting in similar contour dynamics (top right). Major differences in the cell's direction are noticed only at the beginning of the cell track when a stable distribution of virtual markers is not yet reached.

In Figs B.11 and B.12, we present simulated non-polarized and polarized cell tracks for varying temporal resolution $\delta t = 0.25, 0.5, 1, 2, 2.5, 3.\bar{3}s$ and fixed regularization

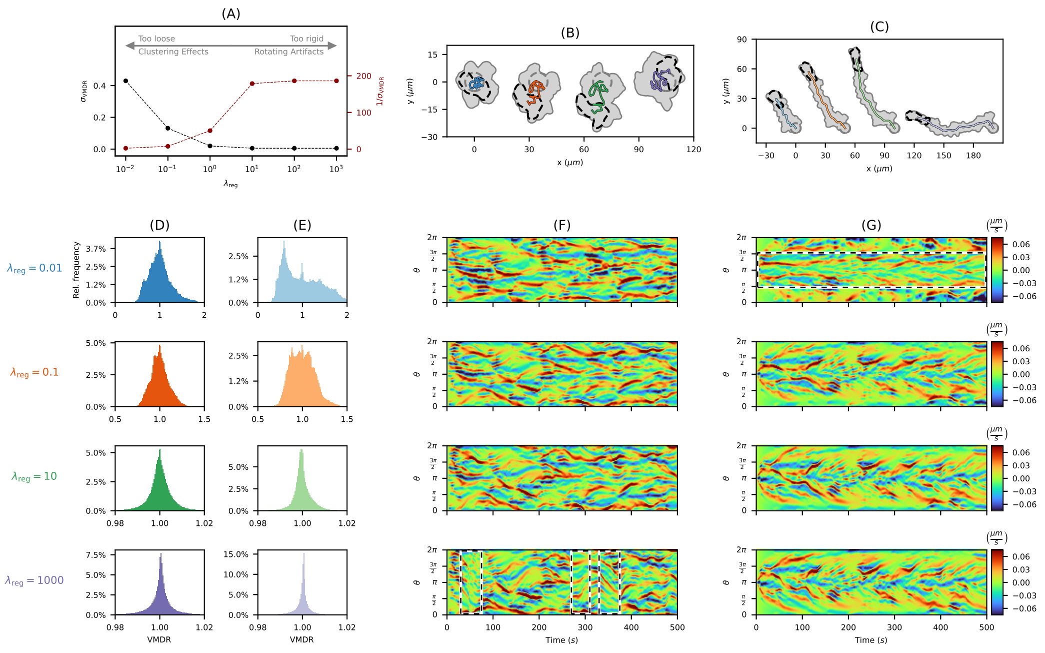


Figure 5.5. **Influence of regularization parameter λ_{reg} on simulated cell tracks. (A)** The impact of λ_{reg} on the virtual marker distribution described by standard deviation of the virtual marker distance ratio σ_{VMDR} and its reciprocal. **(B, C)** Non-polarized (left) and polarized (right) cell tracks generated with varying regularization parameter $\lambda_{\text{reg}} \in \{0.01, 0.1, 10, 1000\}$. The same protrusion component is underlying in **(B)** and **(B)**, respectively. **(D, E)** Histograms of the VMDR for varying λ_{reg} for the non-polarized (left) and polarized (right) cases. **(F, G)** Kymographs of the APCSF component of all non-polarized (left) and polarized (right) cell tracks with regions of interest shown as black and white dashed boxes.

parameter $\lambda_{\text{reg}} = 10$. We observed that the local motion kymograph and hence the general contour dynamics are not substantially affected. In some cases, the overall direction of the contour dynamics was altered due to small differences at the beginning of the track.

In summary, we observed that by varying the regularization parameter within the range $\lambda_{\text{reg}} \in [0.1, 10]$, the overall contour dynamics are barely affected. The same observation was made for a varying temporal resolution. Based on the above studies, we have chosen a medium regularization $\lambda_{\text{reg}} = 10$ and a relatively dense temporal resolution of $\delta t = 0.5s$ for the following simulations.

Long-time simulations are stable and show a normal diffusive behavior To demonstrate the capability of our model to produce stable contour dynamics even for a longer time period, we simulated a variety of cell tracks under both scenarios, the non-polarized and the polarized case as described above. Furthermore, by showing that qualitatively and quantitatively different cell tracks can be generated with our model, we provide a rationale for the inference approach in the following section where we applied the model on experimental cell tracks and different types of locomotion.

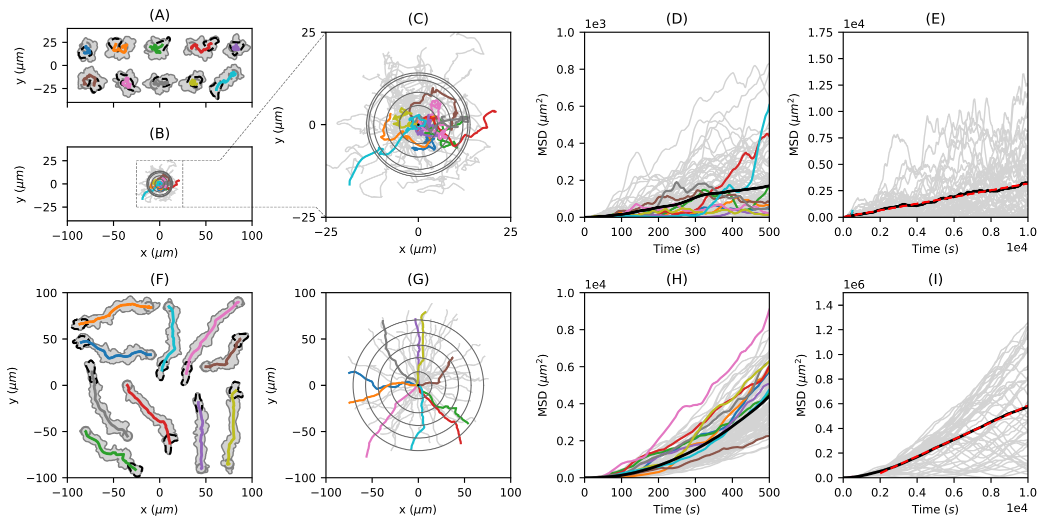


Figure 5.6. Diffusion analysis of artificial cell tracks generated from a non-polarized test scenario (top row) and a polarized test scenario (bottom row). **(A, F)** For each scenario, ten exemplary cell tracks are shown. **(B, C, G)** Center of mass trajectories (colored lines) with root mean squared displacement (grey circles) at different time points with $\Delta t = 100s$. While panel (B) and (G) are scaled equally, panel (C) is an enlarged excerpt of panel (B). **(D, H)** Corresponding MSD of these trajectories (bold black line) for medium time spans of $T = 500s$. **(E, I)** MSD for longer time spans of $T = 10000s$ with linear fit (dashed red line).

In Fig 5.6, we present 50 cell tracks, with 10 especially highlighted (see panels (A) and (F)), for both polarization scenarios. The center of mass trajectories of these cell tracks for $T = 500s$ are displayed in panels (B) and (G). Furthermore, an enlarged excerpt of panel (B) is shown in panel (C). The root mean squared displacements (RMSD) of the trajectories are depicted as gray circles at time steps $t = 0, 100, \ldots, 500s$. The corresponding mean squared displacement (MSD) is shown in panels (D) and (H), indicating normal diffusion for the non-polarized case and a ballistic regime (so-called superdiffusion) for the polarized case for the first 500 seconds. In panels (E) and (I), the MSD is displayed for a longer time period of $T = 10000s$, clearly showing a linear relation between time and MSD indicating normal diffusive behavior also for the polarized case, see also Fig B.8, where the center of mass trajectories of the polarized case are presented for the entire time period of $T = 10000s$. Again, the RMSD is indicated as gray circles ($\delta t = 2000s$). Finally, based on linear fits (red dashed lines), we computed the diffusion coefficients: $D = 0.19\mu m^2/s$ for the non-polarized case and $D = 45.66\mu m^2/s$ for the polarized case.

In Video B.4 and Video B.5, the contour dynamics of the respective cell tracks from Fig 5.6A and F are shown.

5.2.7 Inferring the protrusion component from experimental data

In the first part, we used our model to simulate a variety of different cell tracks based on stable and, in particular, realistic contour dynamics. In the second part, we will now apply our model to experimental data to analyze cell tracks on the individual level and to classify cells based on different types of locomotion: the amoeboid type and a so-called fan-shaped type. The underlying microscopy imaging data was published in [148]. As mentioned earlier, the APCSF and AAF from Eq (5.16) only depend on the current contour and deterministic quantities such as contour curvature, arc length, and area. Hence, the remaining component f_{prot} that propagates one contour to the consecutive one, can be determined explicitly. For a sequence of experimental cell contours, this approach enables us to infer the different model components for a given set of model weights; see Section B.2 for more details on how to estimate these parameters.

In Fig 5.7A, a *D. discoideum* cell track is displayed for a time period of $T = 500s$ and a frame rate of $\delta t = 1s$. This cell track is based on fluorescence microscopy data (see panel (B)), from which the cell contours are segmented (colored lines). As mentioned, we have chosen the 1st percentile of the entire area time series as underlying reference area, i.e., $A_{\text{ref}} = 91.23\mu m^2$. By following the approach from Section B.2, we estimated the following model weights: $w_{\text{prot}} = 6.634\,\mu m^2/s$, $w_{\text{APCSF}} = 0.057\,\mu m^2/s$, and $w_{\text{AAF}} = 3.532\,\mu m/s$. In comparison to parameter values estimated for other cell tracks, see Figs B.13 and B.14, we observed that w_{prot} and w_{AAF} are relatively large while w_{APCSF} took a medium value. This observation coincides with the following cell track characteristics: a fast and persistent movement, an area time series within a smaller range $90\mu m^2 < A < 120\mu m^2$ (see panel (F)), and contour dynamics showing an intermediately regularized curvature. The accuracy of our computational approach is demonstrated in panel (B), with virtual markers (black circles) being effectively propagated onto consecutive contours. In the following, we compare the inferred protrusion component (panel (E)) to commonly used biomarkers: the local motion of the membrane (panel (C)) and the F-actin density/fluorescence intensity close to the membrane (panel (D)).

Inferred protrusion component in comparison to other biomarkers. In contrast to the commonly used local motion which reflects the entire contour dynamics, i.e, protrusions, retractions and minor membrane fluctuations, the proposed model separates the creation of protrusions, which depend on f_{prot}, and retractions, which depend on f_{APCSF} and f_{AAF}. We expect the protrusion component to be correlated to the F-actin density near the membrane. However, here we have chosen images, where the fluorescence signal saturates at high F-actin levels to facilitate segmentation of the cell contours. Thus, by following this approach, the resulting fluorescence intensity of the F-actin density provides less details.

In Fig 5.7, kymographs of all three quantities are displayed: the local motion (panel (C)), the fluorescene intensity (panel (D)), and the inferred protrusion component

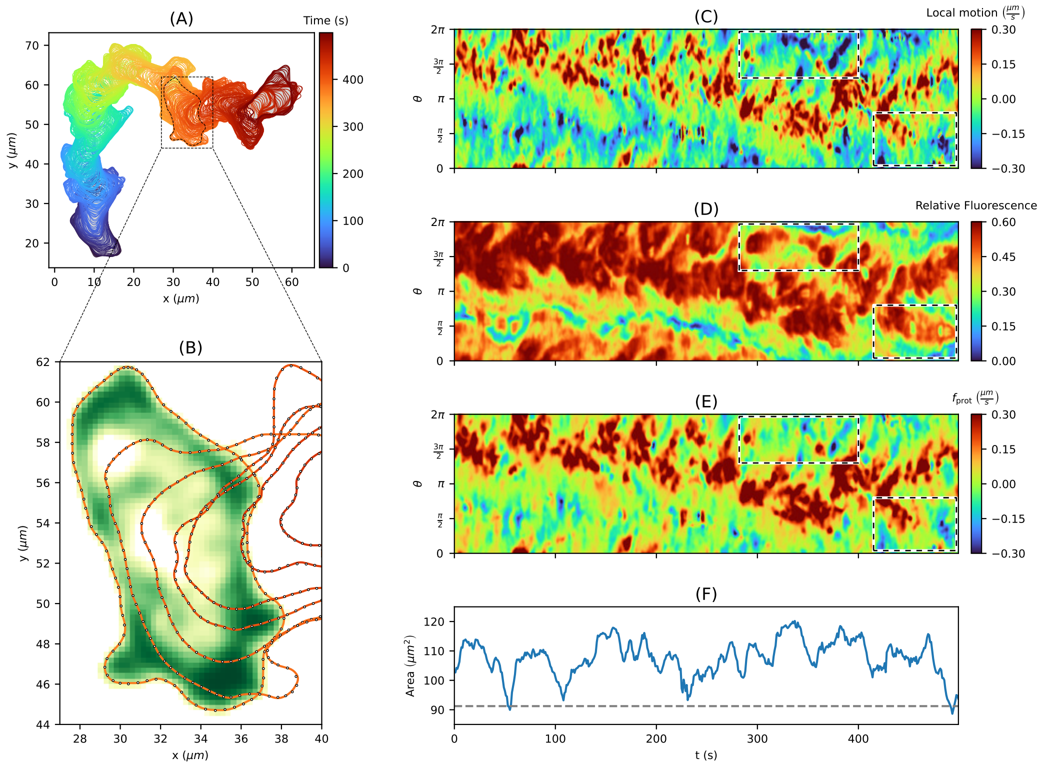


Figure 5.7. Inferring protrusion component from *D. discoideum* cell track. (A) Persistently motile cell track for $T = 500s$. **(B)** Microscopy image with fluorescence intensity (white to green color scheme) and segmented cell contours (red lines, every tenth shown). **(C)** Local motion kymograph showing expansions (red areas) and retractions (blue areas). **(D)** Relative fluorescence intensity with regions of high and low F-actin density displayed in red and blue, respectively. **(E)** The underlying protrusion component inferred from our model for a given set of model parameters. The resulting propagation of virtual markers from one contour to the next one are depicted as black circles in panel (B). **(F)** The contour area with predefined reference area $A_{\text{ref}} = 91.23\mu m^2$ (dashed gray line). Finally, regions of interest are displayed as black and white dashed boxes.

(panel (E)). We expect that a kymograph of a successfully inferred protrusion component would be strictly positive and only indicate regions where protrusions occurred. The remaining retractions are then covered by the other two components of our model: the AAF and the APCSF. In the local motion kymograph, local protrusions (red regions) and retractions (blue regions) are nicely shown, while the fluorescence kymograph captures the main characteristics of the cell motility only. The protrusion component kymograph mainly displays protrusive areas (red regions) with only a very few negative regions (blue). We thus conclude that the missing retractive regions are successfully captured by the AAF and the APCSF.

In Fig B.15, we present our approach to measure the fluorescence intensity near the membrane by averaging over ellipses along the cell contour. To improve the quality of the fluorescence intensity kymograph, we post-processed the recorded microscopy images by using a (tenfold) image upsampling, i.e., we smoothed all microscopy images by increasing the number of pixels tenfold. However, the resulting kymograph showed no differences to the kymograph based on the original image data.

In Fig B.16, the protrusion component from Fig 5.7 is displayed as well as the remaining components f_{APCSF} and f_{AAF}. As the underlying set of model weights, we used the same estimates mentioned above. The contributions of all three model components to the overall dynamics are shown for each time step/contour. In this context, we observed that the APCSF accounts for approximately 10% of the overall dynamics. In contrast, the AAF and the protrusion component are more prominent accounting for approximately 40% and 50%, respectively. This may indicate that the APCSF mainly resolves a smaller number of strongly curved contour segments, while slower contour retractions at the rear of the cell are controlled by the AAF.

In addition, we present kymographs of the same quantities for an alternative set of model parameters, where positive values of f_{prot} are more favored; see Section B.2 for more details. For this case, f_{prot} and f_{AAF} accounted for the overall dynamics equally, whereas f_{APCSF} was observed to be negligible most of the time.

Impact of *a priori* parameter choices on the analysis of experimental data. The F-actin density near the membrane is often interpreted as a marker of protrusive activity. Thus, we would expect some correlation between the F-actin density and the protrusion component in our model. Since the latter is influenced by the choice of the model weights of the two other model components AAF and APCSF, the question arises to what extent the correlation depends on the *a priori* choices of w_{AAF} and w_{APCSF}.

We reformulated our model to be based on relative weights r_{prot}, r_{APCSF}, and $r_{\text{AAF}} = 1 - r_{\text{prot}} - r_{\text{APCSF}}$ and an overall velocity parameter w_f instead of the absolute weights w_{prot}, w_{APCSF}, w_{AAF} used in Eq (5.16); see Section B.4 for more details. In Fig B.17, we display protrusion component kymographs for varying parameters $r_{\text{prot}} \in \{0.05, 0.5, 0.8\}$, $r_{\text{APCSF}} \in \{0.01, 0.05, 0.1\}$, and $w_f \in \{1, 5, 10, 20\}$ of the cell track from Fig 5.7 and the corresponding correlation coefficient between these kymographs and the fluorescence intensity/F-actin density. For a relatively strong APCSF, i.e. $r_{\text{APCSF}} > 0.1$, we observed distinct positive and negative horizontal patches induced by the contour curvature. On the other hand, we obtained predominantly positive values in the kymograph for a strong AAF, e.g., the top left kymograph for $w_f = 20\mu m/s$. Since the AAF affects/shrinks every part of the contour, the counteracting protrusion component is increased to the same amount for the entire contour in order to replicate the given contour dynamics. On the other hand, for a too small AAF, we observed distinct negative (blue) regions in the protrusion component kymograph, indicating that additional retractive forces are necessary to replicate the cell track.

By comparing these protrusion component kymographs with the local motion and the fluorescence intensity from Fig 5.7 panels (C) and (D), we observed the following similarities and differences. In this context, we focus on the two examples highlighted as black and white dashed boxes in each kymograph. The first example ($300s <$

$t < 400s$) clearly indicates a retraction as shown in the local motion kymograph. However, a significant remaining density of F-actin is seen in the fluorescence intensity kymograph. This pattern is also displayed for most of the protrusion component kymographs in Fig B.17, e.g., the top left kymograph on the second page ($w_f = 5\mu m/s$, $r_{\text{prot}} = 0.05$, $r_{\text{APCSF}} = 0.01$). This means that our model predicts a significant amount of f_{prot} necessary to slow down the retraction such that the modeled contour dynamics resemble the experimental data. Without this contribution of the protrusion component, the APCSF and AAF would enforce an even stronger retraction. A similar relation can be observed in the second example ($t > 400s$), but to a lesser extent as in the first example. For all cases, the resulting correlation coefficients between protrusion component and F-actin density fell within a range of $[0.32, 0.48]$. The best fit with a correlation of $\rho = 0.48$ was obtained for multiple parameter choices of $\{w_f, r_{\text{prot}}, r_{\text{APCSF}}\}$ with distinctly different values, e.g., $\{5, 0.05, 0.05\}$, $\{10, 0.8, 0.01\}$, and $\{20, 0.8, 0.01\}$. In case of the estimated (relative) model weights used in Fig 5.7 given by $\{10.2, 0.65, 0.01\}$, we achieved a comparable correlation coefficient of $\rho = 0.47$. Since the correlation coefficient is invariant under changes in location and scale, we conclude that most of the protrusion kymographs displayed in Fig B.17 differ in magnitude primarily. However, the resulting protrusion kymographs show substantial differences with a decreasing correlation coefficient if the APCSF is chosen too strong ($r_{\text{APCSF}} \geq 0.1$). For this reason, the APCSF weight should be chosen relatively low $r_{\text{APCSF}} \leq 0.05$.

In general, we observed that the protrusion component inferred by the model is correlated to the underlying F-actin density. While the protrusion component is affected significantly by the parameter choice, the impact on the above correlation coefficient is minor.

Classification of contour dynamics based on cell motility types. In the first part of this work, we have shown that our model can be used to simulate a variety of different contour dynamics. Here, we demonstrate that our model can also be applied to different experimental cell tracks. By inferring the protrusion component and estimating the underlying model weights, we analyzed contour dynamics on the individual level and classified them based on two different types of locomotion: the amoeboid type and the so-called fan-shaped type. Again, the contour dynamics were derived fluorescence images of *D. discoideum*, where frames are recorded with a temporal resolution of $\delta t = 1s$ (amoeboid type) and $\delta t = 4s$ (fan-shaped type).

The contour parameters $r_{\text{cont}} = 0.6, \sigma_{\text{noise}} = 0.05$ were chosen as in the simulation part, for more information see S1 Text of [DS1]. The reference area A_{ref} was chosen individually for each cell track based on the 1st percentile of the entire time series of the contour area (gray dashed line): 67.14, 128.24, and $69.45\,\mu m^2$ for the amoeboid cell tracks; 149.71, 207.28, and $197.92\,\mu m^2$ for the fan-shaped cells. Moreover, the model weights were estimated for each cell track individually, described as in Section B.2.

In Fig B.13, three tracks of an amoeboid motion as in Fig 5.7 are presented. In contrast to the local motion (first kymograph row), the protrusion component (third kymograph row) contains only a few negative (blue) regions, which means that the APCSF and AAF capture most of the contour retractions successfully. Furthermore, we see strong correlations between all three quantities. However, the protrusive regions in the fluorescence intensity kymographs are larger than for the other two kymographs, which is a direct consequence of detector saturation during fluorescence imaging. From the estimated model weights, we can infer that the third cell track is more motile ($w_{\text{prot}} = 6.505\mu m^2/s$ vs. $w_{\text{prot}} = 4.513\mu m^2/s$) than the other two cell tracks, which coincides with the contour dynamics displayed on top. Furthermore, we see that the AAF weight w_{AAF} directly influences the variability of the area time series, i.e., less variability for the second cell track ($w_{\text{AAF}} = 1.429\mu m/s$) vs. more variability for the first and third cell track ($w_{\text{AAF}} = 0.902\mu m/s$ and $0.676\mu m/s$). Finally, we can observe that the elongated contour shape of the second and third cell track coincide with higher APCSF weights ($w_{\text{APCSF}} = 0.048\mu m^2/s$ and $0.070\mu m^2/s$) compared to the smaller weight of the first cell track ($w_{\text{APCSF}} = 0.023\mu m^2/s$)

In the case of the fan-shaped cells in Fig B.14, we see clear blue stripes in the local motion kymograph at the rear of the cell, which corresponds to a smaller F-actin density in the fluorescence intensity kymograph. Because of the characteristic kidney-shaped cell contour, our model predicts a negative protrusion component at the cell's rear also displayed as a light blue stripe in the kymographs. This indicates that the APCSF and the AAF were not able to fully capture this retraction. Since the APCSF evolves every contour to a circle, it counteracts the characteristic concave-shaped contour of the cell. For this reason, we expect the estimates of w_{APCSF} to be lower than for amoeboid locmotion. Indeed, the APCSF weight was estimated to be zero for all three cells. Therefore, by inferring the impact of the APCSF, we have shown that our model is capable of distinguishing fan-shaped cells from standard amoeboid cells.

For the second and third cell track similar weights $w_{\text{prot}} \approx 4.4\mu m^2/s$ and $w_{\text{AAF}} \approx 1.8\mu m/s$ were estimated which is in accordance with the similar contour dynamics reflected by comparable mean centroid velocities of $0.091\mu m/s$ and $0.099\mu m/s$, respectively. In contrast, we estimated a lower protrusion component weight $w_{\text{prot}} = 3.121\mu m^2/s$ for the first cell track, which is in agreement with a smaller mean centroid velocity of $0.069\mu m/s$. Moreover, we estimated a smaller AAF weight $w_{\text{prot}} = 0.902\mu m^2/s$, which might be a result of the ever-increasing contour area of this track ($130\mu m^2$ to $217\mu m^2$).

5.3 Discussion

We developed a novel model to simulate and analyze the contour dynamics of amoeboid cell migration. The contour dynamics model is based on three components: (1) a stochastic protrusion component based on a self-exciting Poisson point process also know as Hawkes process, (2) an area-preserving curve-shortening flow (APCSF) to regularize the contour arc length, (3) and a further geometric flow introduced as “Area Adjustment Flow” (AAF) to control the contour area. While the first component controls the forward movement of the cell, the latter two components control contour retractions, occurring most often at the rear of the cell.

First, we have shown that our model is capable of generating a variety of cell tracks with different spatio-temporal patterns. We simulated non-polarized as well as polarized contour dynamics that are hardly distinguishable from experimental data and stable over a long time period. Secondly, we applied our model to experimental cell tracks to analyze key motility characteristics based on the inference of the three model components and the estimation of the underlying model component weights. By examining the correlation between the inferred protrusion component and the fluorescence intensity reflecting the F-actin density, we demonstrated that the creation of pseudopods and thus the forward movement of the cell is correctly accounted for by the protrusion component of our model. Furthermore, our model was capable of decoupling the explorative protrusions from the slower contour retractions, which in our model are solely based on the APCSF and the AAF. Finally, by estimating the model weights of each component, we demonstrated a simple approach to classify cells based on two locomotion types: the amoeboid and a so-called fan-shaped type.

The simplest approaches to model amoeboid cell motility focus on the motion of the center of mass of the cell [31, 32, 33]. Often, the center of mass trajectory is modeled by a Fokker-Planck equation [41] or a Langevin equation [10, 30, 11, 32, 31]. Due to a lower model complexity, even a large number of cell tracks can be generated relatively easy and fast. These models are then used to examine statistical quantities such as the diffusion coefficient, the persistence time, and the drift velocity. While our model focuses on the contour dynamics, the center of mass trajectory and the corresponding statistics can be derived easily. For example, we determined a diffusion rate of $D = 0.19\mu m^2/s$ in our test case of simulated non-polarized cell tracks which is in accordance with experimental measurements [149, 41].

Other approaches include biochemical models focusing on intracellular signaling molecules and cytoskeletal components [38, 39] or specific biophysical mechanisms such as cell polarization during chemotaxis [34, 35]. Some models combine intracellular processes with specific membrane patterns such as the formation of pseudopods [36] or changes of the cell shape in general [150]. Furthermore, phase-field models are often used in which the transition of different states such as solid/liquid or interior/exterior of the cell are described [40, 13, 124]. In these models, biochemical markers such as the actin concentration are often propagated in time and space and drive displacements of the cell membrane [14, 75].

In contrast to most of the existing approaches which focus on the microscopic level of chemical and biophysical process during cell migration, our model describes amoeboid motility on a macroscopic level. By relying on the deformation of an elastic object, namely the cell contour, it has a mechanistic basis. Usually, mechanistic approaches are based on the interplay of multiple forces affecting the cell from the outside but also from within [42, 43, 15]. However, some mechanistic models rely on multiple entangled subprocesses making it more difficult to achieve a direct causality between a chosen parameter regime and a specific motility behavior [15]. Here, our model differs substantially due to its dependency on three components only and a low number of parameters: 4 model parameters and 4–5 additional parameters regarding the stochastic process. Desired motility characteristics such as the number, duration, and size of protrusions, the general movement speed or the cell polarization can be easily achieved based on an appropriate specification of the underlying parameter regime.

Motivated by the biological insight that the location of protrusions depends on previous protrusions [6] and is triggered by a cascade of physiological events affecting the growth of the actin filament network [71], we have chosen a Hawkes process as the underlying stochastic process. Due to its self-exciting property, the Hawkes process is capable of generating multiple explorative protrusions in temporal and spatial proximity with subsequent reorientation phases characterized by a temporary decline of the cell motility. Compared to other processes such as the Ornstein-Uhlenbeck process or a standard Poisson process, the Hawkes process is therefore advantageous to model membrane protrusions.

Phase-field models often contains additional constraints to preserve the area/volume of the cell within a specific range [151, 13, 14]. Constraints regarding the surface tension of the membrane are added to these models to regularize the shape or the perimeter of the membrane. In a similar fashion, we use the AAF and APCSF to regularize the contour area and curvature. In contrast to phase-field approaches, the APCSF and AAF affect and propagate a one-dimensional object only, namely the cell contour, in contrast to a two dimensional concentration of biomarkers or even the entire cytoskeleton. Furthermore, phase-field models are successfully used to study more complex processes such as cell division or the interaction and movement of multiple cells [152, 153].

By focusing on the contour dynamics to model cell motility, further assumptions regarding the mapping of virtual markers along time and space were necessary. We assumed that the transport of the underlying protrusion process is based on the concept of regularizing contour flows which was previously introduced in [DS1]. Other approaches to determine contour/virtual marker mappings include electrostatic field equations [28], level-set methods [27, 28], or mechanistic spring equations [29]. All of the above approaches, including ours, share the same problem that the mapping of consecutive contours is not defined *a priori*. We have shown that our contour mapping assumption has a minor effect on the overall contour dynamics and we provided a reasonable range of the underlying regularization parameter λ_{reg} for which simulated contour dynamics are stable.

So far, only few approaches offer a full integration of the numerical model and experimental measurements, e.g., for fibroblast migration [44]. Our model provides a full and automatized integration of experimental measurements due to its capability of inferring protrusion and retraction components individually and by offering a simple and straightforward estimation of the underlying model weights. The most common approach to tune computational models with respect to experimental data is to perform a sensitivity analysis [15, 45, 46, 47]. In this context, different parameter regimes are chosen to simulate different motility behaviors and to determine the importance of each parameter on the simulated outcome. With a similar approach, we have shown that the inference of the protrusion component by our model is stable for varying model weights w_{prot} and w_{AAF}. However, if the APCSF is too prominent, the inferred protrusion component is negatively affected, indicated by a weaker correlation with the underlying F-actin density. Regarding the classification of different motility types, we examined contour dynamics based on a single locomotion type (either amoeboid or fan-shaped). However, recent studies report spontaneous switching between these two locomotion types [DS4].

In summary, the proposed model sets new standards in simulating stable and, in particular, realistic contour dynamics. Due to a fast and straightforward parameter estimation and a fully automatized approach to infer protrusion and retraction characteristics, the model can be used to analyze experimental cell tracks on the individual level and to classify them.

6 Method reproducibility and Open Science approach

In recent years, the issue of a reproducibility crisis in science gained more and more attention [49, 52]. An analysis of 360 research articles in the field of hydrology showed that only a small fraction (1.6%) were fully reproducible [154]. Stagge et al. consider a paper reproducible when (1) a publication states where the associated artifacts can be found, (2) the artifacts are in fact available at the declared location, and (3) the artifacts allow (in large parts) for a reproduction of the results. Furthermore, reproducibility is only ensured when the following so-called primary artifacts are provided: the input data, the corresponding software or source code, and a sufficient documentation to run the source code successfully [154]. In another survey, conducted among German researchers of different disciplines, a majority of participants (55%) stated that the effort to publish research data is high [56]. Noteworthy, only a small fraction of respondents (3%) published at least one data set in the past [56]. Regarding the fields in biophysics and computational biology, a lack of reproducibility often results from erroneous descriptions of the underlying methods, missing documentation to run simulations correctly, or unpublished data and source codes in the first place [155]. These studies underline the necessity to facilitate Open Science methods, to encourage researchers of applying these methods on a regular basis, and to raise awareness to improve reproducibility in science. For this reason, we developed a fully comprehensive Open Science approach as part of the research project "B02 – Inferring the dynamics underlying protrusion-driven cell motility" and the infrastructure project Z03 of the collaborative research center (CRC) 1294 "Data Assimilation".

In Section 6.1.1, we outline the main principles and benefits of Open Science. Then, in Sections 6.1.2 and 6.1.3, we describe the general terminology of research reproducibility and provide further information about common Open Science licenses. In Section 6.2, we present the essential part of our Open Science approach: the development of a `Python`-based open source software package called `AmoePy` to analyze and simulate amoeboid cell motility [DS3]. General information regarding the accessibility, licensing, and documentation of `AmoePy` are provided in Section 6.2.1. The installation process and further instructions to facilitate the usability are described in Section 6.2.2. Then, in Section 6.2.3, we provide details regarding the usage of version control during the development process and we address good scientific practice guidelines with an appropriate archiving of software and research data. Several measures to improve functionality and stability are presented in Section 6.2.4. Finally, in Section 6.3, we conclude with a discussion of our Open Science approach.

6.1 Background and general terminology

First, we clarify certain notions regarding the concepts of Open Science and reproducibility. Furthermore, we describe and compare various licenses, which are commonly used in Open Science approaches.

6.1.1 Open Science principles

In the last years, many definitions have been established to describe the concept of Open Science. In [156], a more general definition is presented: “Open Science is a combination of objective and subjective goals and means to improve science in the diverse subjects and disciplines and as a whole”. A more detailed definition was introduced by Fecher and Friesike who distinguished five different schools of thought: infrastructure, public, democratic, measurement, and pragmatic [157]. These schools target different questions including topics such as the underlying technological architecture, free and equal access, alternative metric systems to measure the impact of research, and easier knowledge transfers in general. However, these schools of thought are not strictly separated and overlap in some areas. Another commonly used approach to define Open Science is based on the following six principles: open access, open data, open source, open methods, open peer review, and open (educational) resources [55]. For a better understanding, we first summarize these principles, see [55] for more details.

Open access. The first principle is the open access to research articles. This includes an access without financial restrictions (“pay walls”) or restrictions based on the geographical location of the researcher (“geoblocking”). Open access accelerates the knowledge transfer of the science community in general, but also brings personal benefits, e.g., higher number of citations, wider scientific outreach in media, and more opportunities regarding funding and potential collaborators [53].

Open data. The second principle is called open data and focuses mainly on the following criteria that research data should fulfill: Findable, Accessible, Interoperable, and Reusable (so-called FAIR principles) [158]. By providing data transparently and well-documented, the reproducibility of research results is strengthened. Furthermore, it can speed up the research process of other researchers, especially, if the process of collecting data was expensive and time-consuming,

Open source. The third principle is called open source and includes open access of source codes which were developed during the research. If external software was used, this software should also be open accessible and without financial restrictions. While open source often focuses on software only, it can also be applied to other fields, e.g, hardware, product, and design blueprints. By following open source standards, the transparency and reproducibility in science is strengthened. Moreover, the researchers benefit through a faster knowledge transfer, but also through the external review of source codes and the underlying methodology.

Open methods. The fourth principle is called open methods and includes the standardization and documentation of scientific methods. By using specific standards regarding, e.g., the experimental setup or the underlying statistical methodology, scientific results can be easier reproduced. Moreover, one can standardize the format of research data to facilitate the distribution and reusability of data. This way, the knowledge transfer can be further enhanced.

Open peer review. The fifth principle is called open peer review and focuses on the peer review process in academia. Proposals to reform the revision process include: revealing reviewers' identities to authors and to the public (open identities), publishing the reviewers' reports alongside the main publication (open reports), and opening the group of reviewers to the research community (open participation). This way, by providing more transparency and a greater scrutiny of the revision process, the trust in science is strengthened.

Open (educational) resources. The last principle is called open resources. It focuses on the teaching of Open Science methods to increase public outreach. Furthermore, it targets the question how Open Science standards can be applied to facilitate the usage of research, software, and data for educational purposes with the aim of improving education in schools and universities globally.

The focus of this work is primarily on open access, open data, and open source, and secondarily on open methods.

6.1.2 What does reproducibility mean?

Similar to the concept of Open Science, multiple definitions are proposed to define reproducibility. In [51], different definitions of reproducibility are presented and, then, classified into different groups. In the following, we apply a definition of reproducibility currently proposed by the Association of Computing Machinery (ACM) [159]; an older version was cited in [51].

Reproducibility. "The measurement can be obtained with stated precision by a different team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using the author's own artifacts." [159].

Additionally to reproducibility (different team, same experimental setup), the following two notions are introduced: repeatability (same team, same experimental setup) and replicability (different team, different experimental setup) [159].

6.1.3 Licenses used in Open Science.

A variety of open/free licenses have been proposed, e.g., the commonly used Creative Commons (CC) licenses, which have been released since 2002 by the non-profit organization of the same name. Based on four basic conditions, six different CC license types have been proposed: CC BY, CC BY-SA, CC BY-NC, CC BY-NC-SA, CC

BY-ND, and CC BY-NC-ND, see [160] for more details. The meaning of each of underlying condition is as follows:

- by attribution (BY), author must be credited in an appropriate way,
- share-alike (SA), work derivatives must be distributed under the same license,
- non-commercial (NC), usage for only non-commercial purposes,
- no derivative (ND), work derivatives are prohibited,

Even less restrictive is the public domain dedication CC0 (“no rights reserved”), which allows adaptation and redistribution without any further conditions. To allow work derivatives and commercial purposes, only CC0, CC BY, and CC BY-SA are considered to be open licenses. Since these licenses do not handle software publications specifically, alternative open source licenses are used often in software development. For example, the GNU General Public License (GNU GPL), released in 1989 by the GNU project, is commonly chosen for software. The current version GPLv3 allows usage, replication, modification, extension, and redistribution under the same license (so-called “copyleft” condition), see [161]. Hence, GNU GPL is comparable to the share-alike license CC BY-SA, while also handling software and patent specific issues.

In [56], a list of simple Open Science practices is presented. For different purposes, they recommend the following licenses: CC-BY-SA for methodologies, GNU GPL for source codes/software, and CERN Open Hardware License for hardware designs. As an alternative to GNU GPL, we recommend so-called permissive licenses, i.e., software licenses without copyleft conditions. For these licenses, the usage and redistribution of the software is permitted even if the derivative software is proprietary. Examples for such permissive licenses include the MIT License, released in 1997 by the Massachusetts Institute of Technology [162], the Apache License (Version 2.0), published in 2004 by the nonprofit Apache Software Foundation [163], and the 2-clause Berkeley Software Distribution (BSD) license [164]. Referring to the latter one, permissive software licenses are also called BSD licenses.

6.2 AmoePy: A `Python`-based toolbox for investigating amoeboid cell motility

In this section, we present AmoePy [DS3]: an open source `Python`-based toolbox to segment cell imaging data, to analyze the resulting contour dynamics as in Chapter 4, and to simulate amoeboid contour dynamics as in Chapter 5. The structure of this section reflects the order of the following working steps to run AmoePy properly. First, in Section 6.2.1, we describe how AmoePy, its code documentation, and further helpful meta files are accessed. In Section 6.2.2, we explain in which ways `AmoePy` can be installed and used. Next, in Section 6.2.3, we describe our usage of version control during the development process and we address good scientific practice guidelines with an appropriate archiving of the software and the underlying research data. Finally, in Section 6.2.4, we present measures to improve the functionality and stability of `AmoePy`: automatic software tests and a hierarchical software structure to ensure correct source code imports within `AmoePy`.

6.2.1 Accessibility, licensing, and understandability

Software packages, especially larger ones, are subject to frequent changes, which may have an impact on the reproducibility of scientific findings. By providing a reset of the source code to a specific date, e.g., right after a journal publication, version control is crucially important to ensure research reproducibility. Due to its built-in version control, the platform `Zenodo` is very suitable for software publications. For the same reason, we published `AmoePy` at `Zenodo`. So far, nine versions (1.0–1.8) of `AmoePy` have been published. Each version can be cited separately based on different digital object identifiers (DOIs). An additional DOI always references the current version of `AmoePy` [DS3]. We have chosen the MIT license [162] to distribute `AmoePy`, because of its focus on software publications and its widespread utilization. In contrast to the commonly used GNU GPL [161], the MIT license is more concise and counts as a so-called permissive software license, i.e, the usage and redistribution of `AmoePy` is permitted even as part of proprietary software.

As proposed in [154], research is only fully reproducible if three so-called primary artifacts are provided: the underlying data, the source code, and instructions to install and run the source code. For this reason, we uploaded the following files as part of our software publication:

- `-README-.md`: README file containing general information and run instructions,
- `-CHEATSHEET-.pdf`: double-sided PDF file (so-called cheat sheet) containing concise instructions regarding installation and usage (see Fig 6.1),
- `B02-AmoePy.zip`: ZIP file containing all source codes, the code documentation, and a small selection of contour data,
- `b02-data.zip`: ZIP file containing the entire contour data used in our research.

In `B02-AmoePy.zip` and `b02-data.zip`, contour data of each cell track are saved in separate TXT files. In [148], we published the primary data as Zeiss LSM files (Laser Scanning Microscope), i.e., the microscopy images from which the secondary contour data was segmented. The primary data for a smaller selection of cell tracks was also published as TIF files at `Zenodo` and can be found within `b02-data.zip`. The file types of our data, i.e, `.txt`, `.tif`, and `.lsm`, are in accordance with the format standardization proposed by the Open Microscopy Environment (OME).

Code documentation. To ensure a high standard of understandability within `AmoePy`, we provide multiple documentation files: a README file, a so-called cheat sheet, and source code documentation automatically generated by the `Python` tool `Sphinx`. The file `-README-.md` contains helpful information and instructions regarding the installation of AmoePy, the usage of the graphical user interface (GUI), and run scripts to reproduce specific research results.

Further information about AmoePy is provided by a double-sided cheat sheet saved as PDF file. In Fig 6.1, both pages of the cheat sheet are displayed. It consists of four groups, highlighted by different background colors with supporting information about:

- the installation process, documentation, dependencies, and settings (gray boxes),
- analysis routines (blue boxes),
- segmentation routines (orange boxes),
- and simulation routines (green boxes).

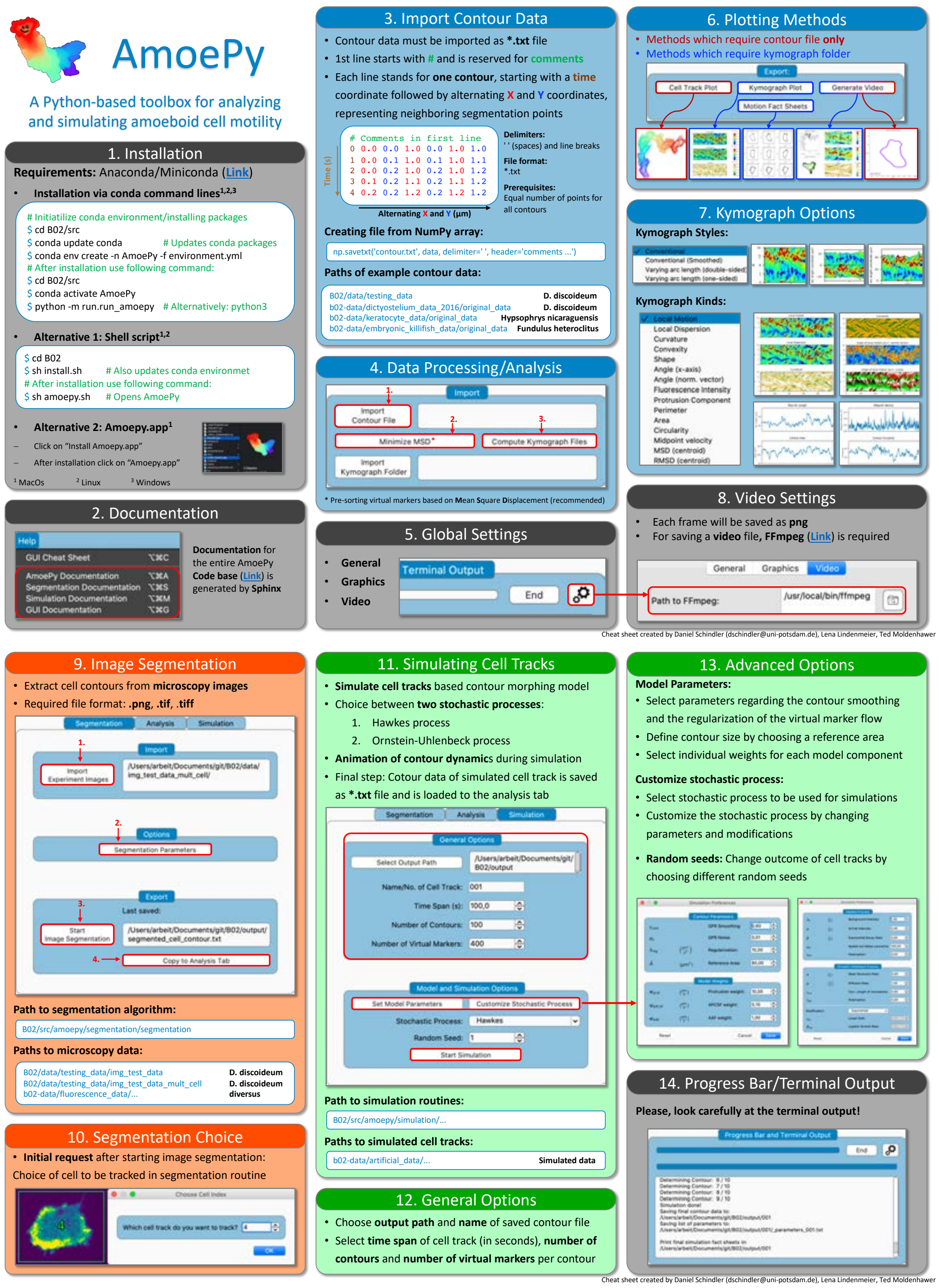
In the first box ("1. Installation"), step-by-step instructions are presented for three different ways to install AmoePy. The other boxes contain helpful instructions when using the GUI of AmoePy. The exact format of the contour data files used within and exported by AmoePy are described in the fourth box ("4. Import Contour Data"). This way, we facilitate the usage of AmoePy with other software packages, e.g., when contour data files were produced with an external image segmentation software but a further analysis within AmoePy is intended.

In addition to the README file and the cheat sheet, we provide a detailed source code documentation, which is automatically generated by Sphinx, a Python-based open source documentation tool. Sphinx converts docstring comments within python source codes into an HTML file, containing descriptions for modules, classes, and methods. Due to additional features, e.g., the inclusion of graphics and mathematical formulas, a visually appealing documentation is generated by Sphinx rapidly and in a fully automated way. By providing a detailed documentation, we enable other researchers to reproduce our results, but also to use AmoePy for their own research. Especially, when using AmoePy without the GUI, a clear and informative source code documentation is crucial. To enhance the readability and consistency of the source code, we followed the PEP8 standard, a collection of guidelines how to write consistent Python code.

In Fig 6.2, an exemplary AmoePy source code is displayed (left) with the corresponding Sphinx code documentation (right). The red boxes in Fig 6.2 illustrate multiple features provided by Sphinx: highlighted paragraphs for hints and warnings (A), mathematical expressions based on LaTeX (B), graphics (C), and tabular descriptions of parameters, input, and output variables (D).

6.2.2 Installation and usability

In this section, we provide further information about the installation process and the underlying dependencies. Subsequently, we present the key aspects regarding the usability of AmoePy: the GUI and additional routines to reproduce the results from [DS1] and [DS2].



AmoePy

A Python-based toolbox for analyzing and simulating amoeboid cell motility

1. Installation

Requirements: Anaconda/Miniconda (Link)

- **Installation via conda command lines**[1,2,3]

```
# Initiatilize conda environment/installing packages
$ cd B02/src
$ conda update conda            # Updates conda packages
$ conda env create -n AmoePy -f environment.yml
# After installation use following command:
$ cd B02/src
$ conda activate AmoePy
$ python -m run.run_amoepy   # Alternatively: python3
```

- **Alternative 1: Shell script**[1,2]

```
$ cd B02
$ sh install.sh        # Also updates conda environmet
# After installation use following command:
$ sh amoepy.sh        # Opens AmoePy
```

- **Alternative 2: Amoepy.app**[1]
  - Click on "Install Amoepy.app"
  - After installation click on "Amoepy.app"

[1] MacOs [2] Linux [3] Windows

2. Documentation

Help	
GUI Cheat Sheet	⌥⌘C
AmoePy Documentation	⌥⌘A
Segmentation Documentation	⌥⌘S
Simulation Documentation	⌥⌘M
GUI Documentation	⌥⌘G

Documentation for the entire AmoePy **Code base** (Link) is generated by **Sphinx**

3. Import Contour Data

- Contour data must be imported as ***.txt** file
- 1st line starts with # and is reserved for **comments**
- Each line stands for **one contour**, starting with a **time** coordinate followed by alternating **X** and **Y** coordinates, representing neighboring segmentation points

```
# Comments in first line
0 0.0 0.0 1.0 0.0 1.0 1.0
1 0.0 0.1 1.0 0.1 1.0 1.1
2 0.0 0.2 1.0 0.2 1.0 1.2
3 0.1 0.2 1.1 0.2 1.1 1.2
4 0.2 0.2 1.2 0.2 1.2 1.2
```

Delimiters: ' ' (spaces) and line breaks
File format: *.txt
Prerequisites: Equal number of points for all contours

Creating file from NumPy array:

```
np.savetxt('contour.txt', data, delimiter=' ', header='comments ...')
```

Paths of example contour data:

B02/data/testing_data	**D. discoideum**
b02-data/dictyostelium_data_2016/original_data	**D. discoideum**
b02-data/keratocyte_data/original_data	**Hypsophrys nicaraguensis**
b02-data/embryonic_killifish_data/original_data	**Fundulus heteroclitus**

4. Data Processing/Analysis

* Pre-sorting virtual markers based on **M**ean **S**quare **D**isplacement (recommended)

5. Global Settings

- **General**
- **Graphics**
- **Video**

6. Plotting Methods

- Methods which require contour file **only**
- Methods which require kymograph folder

7. Kymograph Options

Kymograph Styles:

✓ Conventional
Conventional (Smoothed)
Varying arc length (double-sided)
Varying arc length (one-sided)

Kymograph Kinds:

8. Video Settings

- Each frame will be saved as **png**
- For saving a **video** file, **FFmpeg** (Link) is required

Path to FFmpeg: /usr/local/bin/ffmpeg

Cheat sheet created by Daniel Schindler (dschindler@uni-potsdam.de), Lena Lindenmeier, Ted Moldenhawer

9. Image Segmentation

- Extract cell contours from **microscopy images**
- Required file format: **.png**, **.tif**, **.tiff**

/Users/arbeit/Documents/git/B02/data/img_test_data_mult_cell/

Last saved: /Users/arbeit/Documents/git/B02/output/segmented_cell_contour.txt

Path to segmentation algorithm:

B02/src/amoepy/segmentation/segmentation

Paths to microscopy data:

B02/data/testing_data/img_test_data	**D. discoideum**
B02/data/testing_data/img_test_data_mult_cell	**D. discoideum**
b02-data/fluorescence_data/...	**diversus**

10. Segmentation Choice

- **Initial request** after starting image segmentation:

Choice of cell to be tracked in segmentation routine

Which cell track do you want to track? 4

11. Simulating Cell Tracks

- **Simulate cell tracks** based contour morphing model
- Choice between **two stochastic processes**:
  1. Hawkes process
  2. Ornstein-Uhlenbeck process
- **Animation of contour dynamic**s during simulation
- Final step: Cotour data of simulated cell track is saved as ***.txt** file and is loaded to the analysis tab

/Users/arbeit/Documents/git/B02/output
Name/No. of Cell Track: 001
Time Span (s): 100,0
Number of Contours: 100
Number of Virtual Markers: 400
Stochastic Process: Hawkes
Random Seed: 1

Path to simulation routines:

B02/src/amoepy/simulation/...

Paths to simulated cell tracks:

b02-data/artificial_data/...	**Simulated data**

12. General Options

- Choose **output path** and **name** of saved contour file
- Select **time span** of cell track (in seconds), **number of contours** and **number of virtual markers** per contour

13. Advanced Options

Model Parameters:

- Select parameters regarding the contour smoothing and the regularization of the virtual marker flow
- Define contour size by choosing a reference area
- Select individual weights for each model component

Customize stochastic process:

- Select stochastic process to be used for simulations
- Customize the stochastic process by changing parameters and modifications
- **Random seeds:** Change outcome of cell tracks by choosing different random seeds

14. Progress Bar/Terminal Output

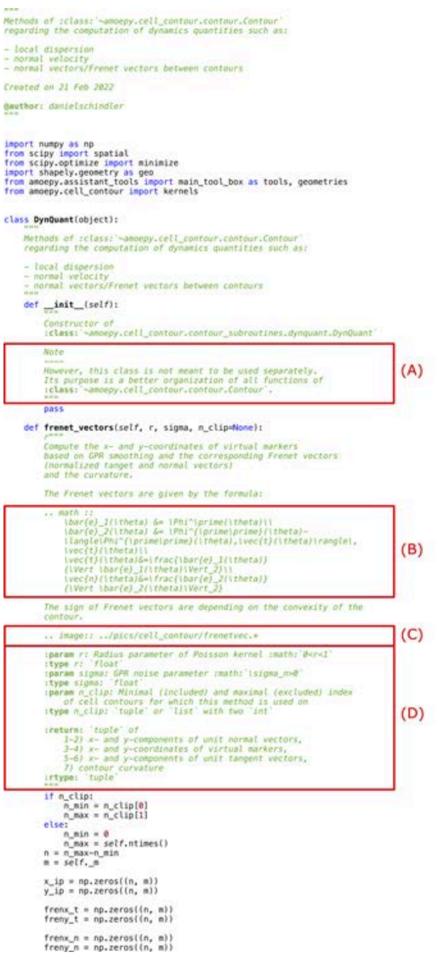
Please, look carefully at the terminal output!

```
Determining Contour:  6 / 10
Determining Contour:  7 / 10
Determining Contour:  8 / 10
Determining Contour:  9 / 10
Simulation done!
Saving final contour data to:
/Users/arbeit/Documents/git/B02/output/001
Saving list of parameters to:
/Users/arbeit/Documents/git/B02/output/001/_parameters_001.txt

Print final simulation fact sheets in:
/Users/arbeit/Documents/git/B02/output/001
```

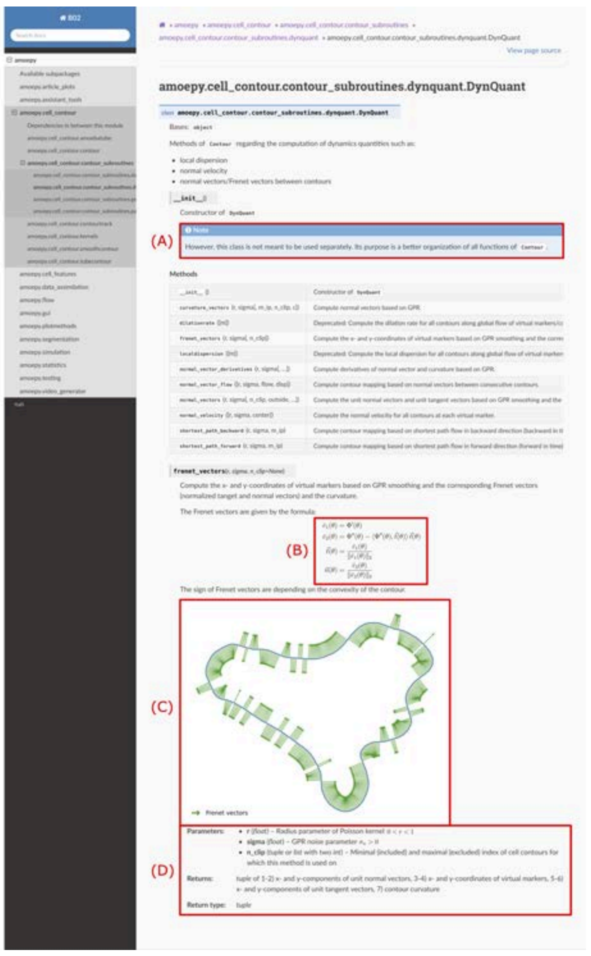
Cheat sheet created by Daniel Schindler (dschindler@uni-potsdam.de), Lena Lindenmeier, Ted Moldenhawer

Figure 6.1. AmoePy **cheat sheet** containing general information about installing AmoePy, the code documentation, and settings within the GUI (gray boxes). Furthermore, it provides information about specific routines regarding the analysis (blue boxes), the segmentation (orange boxes), and the simulation (green boxes) of contour dynamics.

Installation and dependencies. We have chosen Python as underlying programming language for a number of reasons. First, Python offers a variety of advanced packages for different scientific purposes, e.g., numerical analysis (NumPy and SciPy), geometry (shapely), image processing (scikit-image), visualization (matplotlib), data



```
"""
Methods of :class:`~amoepy.cell_contour.contour.Contour`
regarding the computation of dynamics quantities such as:

- local dispersion
- normal velocity
- normal vectors/Frenet vectors between contours

Created on 21 Feb 2022

@author: danielschindler
"""


import numpy as np
from scipy import spatial
from scipy.optimize import minimize
import shapely.geometry as geo
from amoepy.assistant_tools import main_tool_box as tools, geometries
from amoepy.cell_contour import kernels


class DynQuant(object):
    """
    Methods of :class:`~amoepy.cell_contour.contour.Contour`
    regarding the computation of dynamics quantities such as:

    - local dispersion
    - normal velocity
    - normal vectors/Frenet vectors between contours
    """
    def __init__(self):
        """
        Constructor of
        :class:`~amoepy.cell_contour.contour_subroutines.dynquant.DynQuant`

        Note
        ----
        However, this class is not meant to be used separately.
        Its purpose is a better organization of all functions of
        :class:`~amoepy.cell_contour.contour.Contour`.
        """
        pass

    def frenet_vectors(self, r, sigma, n_clip=None):
        r"""
        Compute the x- and y-coordinates of virtual markers
        based on GPR smoothing and the corresponding Frenet vectors
        (normalized tanget and normal vectors)
        and the curvature.

        The Frenet vectors are given by the formula:

        .. math ::
            \bar{e}_1(\theta) &= \Phi^\prime(\theta)\\
            \bar{e}_2(\theta) &= \Phi^{\prime\prime}(\theta)-
            \langle\Phi^{\prime\prime}(\theta),\vec{t}(\theta)\rangle\,
            \vec{t}(\theta)\\
            \vec{t}(\theta)&=\frac{\bar{e}_1(\theta)}
            {\Vert \bar{e}_1(\theta)\Vert_2}\\
            \vec{n}(\theta)&=\frac{\bar{e}_2(\theta)}
            {\Vert \bar{e}_2(\theta)\Vert_2}

        The sign of Frenet vectors are depending on the convexity of the
        contour.

        .. image:: ../pics/cell_contour/frenetvec.*

        :param r: Radius parameter of Poisson kernel :math:`0<r<1`
        :type r: `float`
        :param sigma: GPR noise parameter :math:`\sigma_n>0`
        :type sigma: `float`
        :param n_clip: Minimal (included) and maximal (excluded) index
            of cell contours for which this method is used on
        :type n_clip: `tuple` or `list` with two `int`

        :return: `tuple` of
            1-2) x- and y-components of unit normal vectors,
            3-4) x- and y-coordinates of virtual markers,
            5-6) x- and y-components of unit tangent vectors,
            7) contour curvature
        :rtype: `tuple`
        """
        if n_clip:
            n_min = n_clip[0]
            n_max = n_clip[1]
        else:
            n_min = 0
            n_max = self.ntimes()
        n = n_max-n_min
        m = self._m

        x_ip = np.zeros((n, m))
        y_ip = np.zeros((n, m))

        frenx_t = np.zeros((n, m))
        freny_t = np.zeros((n, m))

        frenx_n = np.zeros((n, m))
        freny_n = np.zeros((n, m))
```



» amoepy » amoepy.cell_contour » amoepy.cell_contour.contour_subroutines » amoepy.cell_contour.contour_subroutines.dynquant » amoepy.cell_contour.contour_subroutines.dynquant.DynQuant

View page source

amoepy.cell_contour.contour_subroutines.dynquant.DynQuant

class `amoepy.cell_contour.contour_subroutines.dynquant.DynQuant`

Bases: `object`

Methods of `Contour` regarding the computation of dynamics quantities such as:

- local dispersion
- normal velocity
- normal vectors/Frenet vectors between contours

`__init__()`

Constructor of `DynQuant`

Note

However, this class is not meant to be used separately. Its purpose is a better organization of all functions of `Contour`.

Methods

`__init__`()	Constructor of `DynQuant`
`curvature_vectors`(r, sigma[, m_ip, n_clip, c])	Compute normal vectors based on GPR.
`dilationrate`([n])	Deprecated: Compute the dilation rate for all contours along global flow of virtual markers/c
`frenet_vectors`(r, sigma[, n_clip])	Compute the x- and y-coordinates of virtual markers based on GPR smoothing and the corre
`localdispersion`([n])	Deprecated: Compute the local dispersion for all contours along global flow of virtual marker
`normal_vector_derivatives`(r, sigma[, ...])	Compute derivatives of normal vector and curvature based on GPR.
`normal_vector_flow`(r, sigma, flow, disp)	Compute contour mapping based on normal vectors between consecutive contours.
`normal_vectors`(r, sigma[, n_clip, outside, ...])	Compute the unit normal vectors and unit tangent vectors based on GPR smoothing and the
`normal_velocity`(r, sigma, center)	Compute the normal velocity for all contours at each virtual marker.
`shortest_path_backward`(r, sigma, m_ip)	Compute contour mapping based on shortest path flow in backward direction (backward in ti
`shortest_path_forward`(r, sigma, m_ip)	Compute contour mapping based on shortest path flow in forward direction (forward in time)

`frenet_vectors`(r, sigma, n_clip=None)

Compute the x- and y-coordinates of virtual markers based on GPR smoothing and the corresponding Frenet vectors (normalized tangent and normal vectors) and the curvature.

The Frenet vectors are given by the formula:

$$\begin{aligned}\bar{e}_1(\theta) &= \Phi'(\theta)\\ \bar{e}_2(\theta) &= \Phi''(\theta) - \langle\Phi''(\theta), \vec{t}(\theta)\rangle\, \vec{t}(\theta)\\ \vec{t}(\theta) &= \frac{\bar{e}_1(\theta)}{\Vert\bar{e}_1(\theta)\Vert_2}\\ \vec{n}(\theta) &= \frac{\bar{e}_2(\theta)}{\Vert\bar{e}_2(\theta)\Vert_2}\end{aligned}$$

The sign of Frenet vectors are depending on the convexity of the contour.

Parameters:	• r (float) – Radius parameter of Poisson kernel $0 < r < 1$ • sigma (float) – GPR noise parameter $\sigma_n > 0$ • n_clip (tuple or list with two int) – Minimal (included) and maximal (excluded) index of cell contours for which this method is used on
Returns:	tuple of 1-2) x- and y-components of unit normal vectors, 3-4) x- and y-coordinates of virtual markers, 5-6) x- and y-components of unit tangent vectors, 7) contour curvature
Return type:	tuple

Figure 6.2. **Code documentation generated by `Sphinx`.** Based on the `Python` source code (left), the corresponding code documentation is automatically generated by `Sphinx` and saved as HTML file. The red boxes illustrate several features provided by `Sphinx`: paragraphs for highlighted hints or warnings **(A)**, `LaTeX` formulas **(B)**, graphics **(C)**, and tabular parameter descriptions **(D)**.

analysis (`pandas`), and GUI development (`PyQt5`). Thus, `Python` provides an all-in-one solution for our multidisciplinary application. Secondly, `Python` and all packages used within `AmoePy` rely on large developer communities and are distributed under open source software licenses, e.g., GNU GPLv3, CC-BY, and BSD licenses. By avoiding external commercial software or programming languages, e.g., `Matlab`, our software package is in accordance with the open source principles previously described in Section 6.1.1.

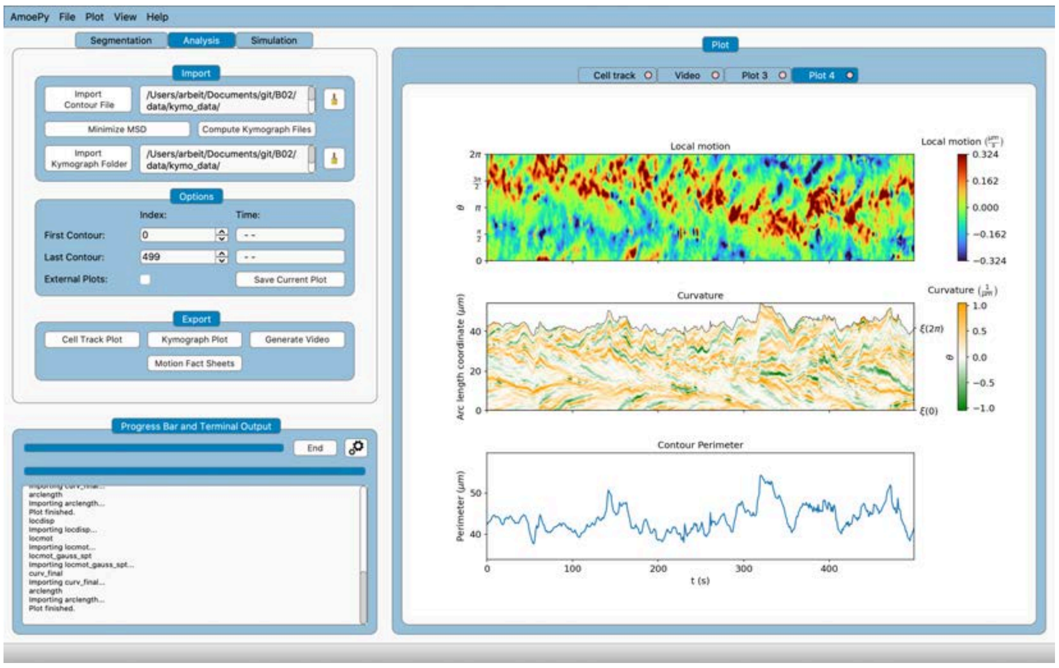
To facilitate the installation process regarding the download of required `Python` packages, we used `Conda`, an open source package manager for `Python`. Based on a YML file, which consists of a list of all required packages and their exact version numbers, a `Conda` environment is created, see Fig 6.1 ("1. Installation"). Afterwards, all package dependencies are managed and downloaded via `Conda` automatically, making the installation process fast, easy, and platform-independent.

Graphical user interface (GUI). Currently, we provide three different ways to start the GUI of `AmoePy`, see Fig 6.1 ("1. Installation"). First, the GUI can be opened by running a python script in the `Python` command line interface (CLI). Second, we provide shell scripts to install `AmoePy` and to start the GUI. Third, we provide an executable `macOS` application `Amoepy.app`, which enables a simple and user-friendly way to run `AmoePy`. However, we recommend the first option due to the platform independence of the python script.

In Fig 6.3, a screenshot of the `AmoePy` GUI is shown. The GUI consists of a menu bar, which contains additional routines and options, and three main windows: a processing window, a progress window, and a plotting window. The processing window (top left) contains all routines regarding the segmentation, analysis, and simulation of contour dynamics. The progress window (bottom left) contains a progress bar and a text output, providing information during each processing routine. On the right-hand side, the plotting window is positioned, which contains different visualization types, e.g., cell tracks, animated contour dynamics, or kymographs.

The main purpose of the GUI is to further facilitate the usage of `AmoePy`, especially, for researchers with no profound `Python` knowledge. By providing an easy-to-use GUI, other target groups, e.g., biologists and biophysicists, are encouraged to apply the advanced mathematical methods of [DS1] and [DS2], but without spending much time in code development. This way, the outreach of our research is potentially increased. One important goal of our work was to develop an all-in-one software package. For this reason, `AmoePy` allows for image segmentation, in which a sequence of microscopy images saved as PNG or TIF files is translated into a TXT file containing the contour dynamics, see Fig 6.1 (orange boxes). Subsequently, this TXT file is loaded to the analysis tab of the processing window. Corresponding quantities for these contour dynamics, e.g., local motion, curvature, arc length, can then be determined and further visualized as well as the underlying cell track, see Fig 6.1 (blue boxes). Alternatively, contour dynamics can be generated without experimental recordings/microscopy data, by using our model from Chapter 5, see Fig 6.1 (green boxes). Every processing routine included in the GUI can be also accessed via the default `Python` CLI. When using `AmoePy` without the GUI, e.g., to further customize specific functions or to reduce the computation time, we highly recommend to read the source code documentation as displayed in Fig 6.2.

Reproducing research results. The first purpose of `AmoePy` is to ensure the accessibility and usability of the underlying mathematical methods for other researchers, e.g., by providing an easy-to-use GUI. The second purpose is to improve the reproducibility of our own research results. As described earlier, Stagge et al. consider research work to be fully reproducible only if the following three prerequisites are provided: the underlying data, the source codes, and instructions to run the source code [154]. In [148] and [DS3], the primary data, i.e., the microscopy imaging data, and the secondary data, i.e., the cell contour files, are provided, respectively. In [DS3], we published all necessary source codes and a detailed documentation of the source code in form of an HTML file generated by `Sphinx` and the cheat sheet shown in



```
/Users/arbeit/Documents/git/B02/data/kymo_data/
/Users/arbeit/Documents/git/B02/data/kymo_data/
First Contour: 0
Last Contour: 499
```

```
importing curv_final...
arclength
importing arclength...
Plot finished.
locdisp
importing locdisp...
locmot
importing locmot...
locmot_gauss_spt
importing locmot_gauss_spt...
curv_final
importing curv_final...
arclength
importing arclength...
Plot finished.
```

Figure 6.3. **graphical user interface (GUI) of `AmoePy`.** The GUI contains a menu bar and three main windows. The top left window provides all processing routines regarding segmentation, analysis, and simulation of contour dynamics. In the left bottom corner, a progress of each routine is shown as progress bar and text output. In the window on the right-hand side, all plots and videos are shown.

Fig 6.2. Furthermore, our research articles [DS1] and [DS2] provide additional pseudocodes to further facilitate the reimplementation of our mathematical methods. However, these documents do not yet ensure a self-explanatory usage of `AmoePy` to reproduce the research results from [DS1] and [DS2]. For this reason, we provide two python scripts, one for each article, with which all figures, statistics and resulting contour files are recreated. Full instructions, containing single command lines to run each script, can be found within the README file. By providing all three primary artifacts, described as in [154], we enable other researcher to fully reproduce the results of our research articles.

6.2.3 Archiving and extensibility

The source code underlying our research work was collectively developed and secured by using `GitLab`, an open source version control software based on `Git`. In contrast to the nine versions of `AmoePy`, published on `Zenodo` over a period of three years, we used `GitLab` on a daily to weekly basis to secure the current software progress. Due to multiple improvements of the underlying methods and algorithms, changes of resulting contour data and kymograph files were inevitable and, therefore, difficult to handle via `Git`. Even for minor numerical deviations, `Git` proposed to rewrite and backup these files completely, leading to a bloated repository size. For this reason, we installed `Git LFS` (large file storage), an open source extension of `Git` to manage large files and, in particular, binary files, e.g., videos and images of cell tracks. By using `Git LFS`, selected (large) files are saved in an additional repository, while a copy of the current file version and a pointer file to that copy remain on the local repository. However, `Git LFS` does not clean up the potentially large history of an already existing repository. We, therefore, recommend to install `Git LFS` when starting a new research project.

According to the good scientific practice guidelines of the Deutsche Forschungsgemeinschaft (DFG), research data should be archived in an "accessible and identifiable manner for a period of ten years at the institution where the data were produced" [165]. For this reason, we used `DSpace`, an open source repository software with a focus on data archiving. After each article publication, we uploaded the underlying data, source codes, instructions, and other meta data on a university server. By doing so, we further enable the reproducibility of our research and allow for extensibility, i.e., the possibility for other researchers to extend our software in future projects. Moreover, our approach is in accordance with another commonly applied standard in the field of data archiving, the so-called 3-2-1 rule, i.e., three data copies are secured on at least two different storage types with one copy being saved remotely.

6.2.4 Functionality and stability

In the course of this project, we took several measures to ensure and improve the functionality and stability of `AmoePy`. First, we implemented automatic software tests by using the continuous integration/continuous delivery (CI/CD) interface of `Gitlab` and the `Python` testing framework `Pytest`. Secondly, to improve the stability and the clarity of the source code, we analyzed all import dependencies within `AmoePy` by using a `Python` package called `Pydeps`. In this context, we created a top-down hierarchy of all `AmoePy` sub-packages and resolved circular imports/dependencies, e.g., two scripts which import each other.

Automatic software testing. In software development, CI/CD is a widely-used framework to automatize (1) the integration of source code changes into the main branch of an existing repository and (2) the delivery/deployment of the repository to each user. By using CI/CD, software tests can be started automatically, e.g, after every commit to the repository or with respect to predefined schedules. If these tests are not successful, notifications are sent automatically to the repository maintainers. In such cases, we either made further changes to the source code to fix all tests or we have chosen to reset the repository to a previous version where no test failures occurred. With regards to our research, the main benefits of using CI/CD are potential time savings due to the automation of test routines and the early detection of erroneous parts of the source code.

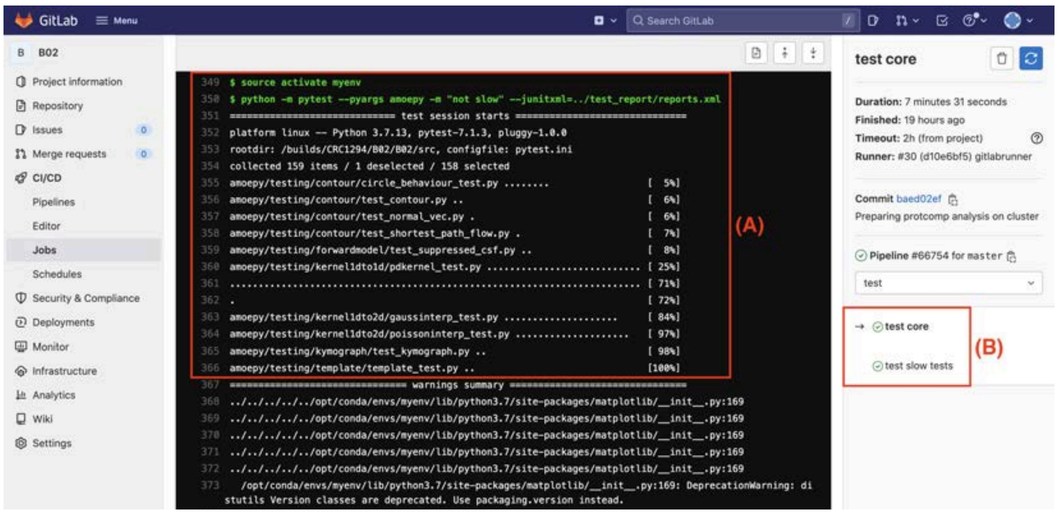
We implemented different kinds of automatic tests in `AmoePy`: unit tests, regression tests, and deployment tests. With unit tests, small individual functions (so-called units) are checked for correctness. In this context, we implemented tests to verify specific functions regarding the computation of several geometric quantities of contours (area, arc length, and curvature). For circles and ellipses of different sizes, we compared the numerical results with the analytical solutions of these quantities. Then, based on predefined error thresholds, each unit test either passed or failed.

The second group are called regression tests, where we defined a small number of end-to-end scenarios (from data to results), e.g., the computation of contour mappings and the resulting kymograph quantities (local motion, local dispersion). In contrast to the unit tests, where the output was compared to a true reference value, there is no true or false answer for most of these regression tests. However, regression tests play an equally important role to ensure proper stability of the software. By implementing different regression tests, we checked if the source code is executed successfully from start to end. This way, the interplay of different parts of the source code and the correct passing of input/output variables between them is monitored. With `Pytest`, a `Python` testing framework, we selected which routines in `AmoePy` are run as automatic tests. Furthermore, by using `Pytest` and `Gitlab`, we defined two testing scenarios/pipelines based on the computation time of each routine. We excluded regression tests with long computation times to a pipeline called "slow tests", which was scheduled to start at 4 a.m. on every day. The remaining functions are tested after every commit in a pipeline called "core", for which we defined a collective time-out failure after two hours.

As a final step, we implemented deployment tests to check if all required python packages are downloaded correctly during the installation process. Additionally, we used deployment tests to check if the automatic source code documentation is generated without errors or warnings. By adding these deployment tests to the CI/CD pipeline, we ensure that every software version is provided with an accurate source code documentation and that the underlying `conda` environment, necessary to run AmoePy, is correctly installed.

In Fig 6.4, a screenshot of the browser interface of `Gitlab` is presented. Based on a `Python` command line, a collection of 158 tests from 10 different files was tested (panel (A)). For each of these files, the file path and the number of individual tests, represented by a sequence of dots, are displayed. In panel (B), we see that that the "core" pipeline, instead of the "slow test" pipeline, is selected. The green check marks indicate that both pipelines finished successfully.

Data validation tests. In addition to the above tests, which are executed automatically on a remote server, we implemented several data validation tests. When a contour data file is loaded to `AmoePy`, these tests are started internally and perform various format checks, e.g, that the file contains at least one contour with three segmentation points and that each contour possess the same number of segmentation points. In the newest version of `AmoePy`, a varying number of segmentation points per contour is accepted. Based on the highest number of segmentation points of a single contour, the number of segmentation points for the other contour is adjusted. Then, by using a GPR for each contour separately, segmentation points are added, if necessary, to the contour and redistributed along the cell contour in an equidistant manner. The last point arose from feedback by an external researcher in the field of cell



```
$ source activate myenv
$ python -m pytest --pyargs amoepy -m "not slow" --junitxml=../test_report/reports.xml
============================= test session starts ==============================
platform linux -- Python 3.7.13, pytest-7.1.3, pluggy-1.0.0
rootdir: /builds/CRC1294/B02/B02/src, configfile: pytest.ini
collected 159 items / 1 deselected / 158 selected
amoepy/testing/contour/circle_behaviour_test.py ........                  [  5%]
amoepy/testing/contour/test_contour.py ..                                 [  6%]
amoepy/testing/contour/test_normal_vec.py .                               [  6%]
amoepy/testing/contour/test_shortest_path_flow.py .                       [  7%]
amoepy/testing/forwardmodel/test_suppressed_csf.py ..                     [  8%]
amoepy/testing/kernel1dto1d/pdkernel_test.py ........................... [ 25%]
........................................................................ [ 71%]
.                                                                        [ 72%]
amoepy/testing/kernel1dto2d/gaussinterp_test.py ....................    [ 84%]
amoepy/testing/kernel1dto2d/poissoninterp_test.py ....................  [ 97%]
amoepy/testing/kymograph/test_kymograph.py ..                            [ 98%]
amoepy/testing/template/template_test.py ..                              [100%]
=============================== warnings summary ===============================
../../../../../opt/conda/envs/myenv/lib/python3.7/site-packages/matplotlib/__init__.py:169
../../../../../opt/conda/envs/myenv/lib/python3.7/site-packages/matplotlib/__init__.py:169
../../../../../opt/conda/envs/myenv/lib/python3.7/site-packages/matplotlib/__init__.py:169
../../../../../opt/conda/envs/myenv/lib/python3.7/site-packages/matplotlib/__init__.py:169
../../../../../opt/conda/envs/myenv/lib/python3.7/site-packages/matplotlib/__init__.py:169
  /opt/conda/envs/myenv/lib/python3.7/site-packages/matplotlib/__init__.py:169: DeprecationWarning: di
stutils Version classes are deprecated. Use packaging.version instead.
```

Duration: 7 minutes 31 seconds
Finished: 19 hours ago
Timeout: 2h (from project)
Runner: #30 (d10e6bf5) gitlabrunner

Commit baed02ef
Preparing protcomp analysis on cluster

Pipeline #66754 for master

Figure 6.4. `Gitlab` **interface with CI/CD pipeline, (A)** Text output showing that a collection of 158 automatic tests was successfully executed from start to end. **(B)** Two different scenarios/pipelines are defined: "core" (currently selected) and "slow tests".

migration, who intended to use AmoePy to analyze cell contour data. However, the contour data was produced with an external image segmentation software, which allowed for a varying number of segmentation points per contour. After committing the above changes, the researcher was able to further process the contour data within AmoePy. This example shows, how software publications can accelerate the knowledge transfer and increase the own scientific outreach.

Internal software structure. In [DS1], the source code is found in two directories: '`B02/src/amoepy`' and '`B02/src/run`'. The first directory contains the AmoePy software package, while the second directory provides additional run scripts, e.g., to start the GUI and to reproduce results and figures of the articles [DS1] and [DS2]. To better organize all Python scripts within AmoePy, we used the built-in Python package structure. Based on specific files named '`__init__.py`', which are saved into the root directory and all sub-directories of the software package, an underlying hierarchy is established. This structure is then used to easily manage source code imports from other files and directories. To analyze and illustrate the import dependency structure of AmoePy, we used an open source `Python` package called `Pydeps`.

In Fig 6.5A, `Pydeps` was used to analyze import dependencies between the first level sub-packages of AmoePy. All sub-packages are positioned based on their purposes: either general (top) or more specialized (bottom). The arrows between these sub-packages represent import dependencies, e.g., '`assistant_tools`' (purple), which is used in every other sub-package. In contrast, the sub-packages '`gui`' and '`article_plots`' (dark red, bottom) depend on almost all other sub-packages and are not inherited further. The color of each sub-package corresponds to its hierarchical position in AmoePy. Due to the clear organization of source code files and a strict top-bottom hierarchy, routines for specific purposes, e.g., segmentation, simulation, or analysis, can now be found easier, which further increases the usability

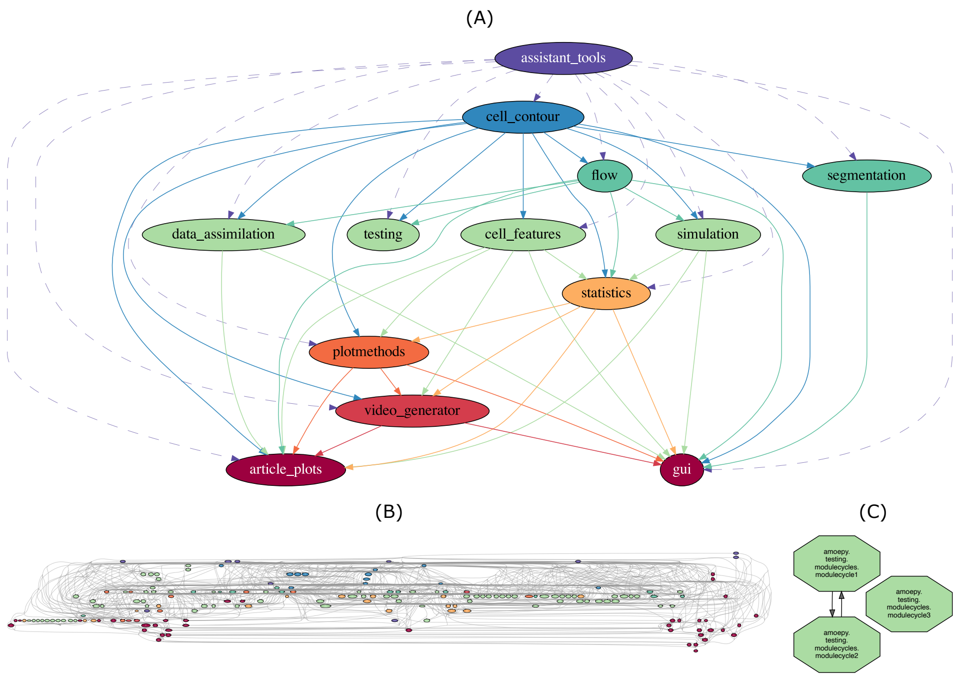


Figure 6.5. Module/package structure within AmoePy and import dependencies. (A) Sub-packages of AmoePy with top-down hierarchy, i.e., the position/color of each package represents its purpose ranging from general (top) to specialized (bottom). Dependencies of the sup-package 'assistant_tools' (purple) are shown as dashed arrows, due to its usage in every other sub-package. **(B)** Import dependency graph of all files within AmoePy. The color of each file corresponds to the hierarchy level of the affiliated sub-package. **(C)** Reduced dependency graph showing circular dependencies only. For testing reasons, two cycles were produced: two files, which import each other (left), and one file, which imports itself (right).

and extensibility of AmoePy. This graph is also provided in the beginning of the source code documentation.

We analyzed the package structure of AmoePy to rule out possible cross or circular dependencies. Such dependencies pose a potential threat to the software stability by producing failures due to an infinite recursion of package imports. In panel (A), we see that cross and circular import dependencies do not occur in the first level of sub-packages. However, they may still occur within these sub-packages, e.g., two files in the same sub-directory which import each other. For this reason, the detection of possible cross/circular dependencies was also performed on the level of individual source code files. In panel (B), the dependency graph is illustrated, which contains all 177 source code files within AmoePy. Based on a Pydeps analysis, this graph does not contain any cross or circular dependencies except for two test cases: two files importing each other and one file importing itself, see panel (C).

6.3 Discussion

In this chapter, we complemented the results from [DS1] and [DS2] with a fully comprehensive Open Science approach. The main aspects of this approach are a primary data publication [148] and a secondary data plus software publication [DS3].

The software publication accomplishes two main goals: (1) to support research reproducibility of the above articles and (2) to provide all mathematical models in our easy-to-use and well-documented software package `AmoePy`. More precisely, to further improve its usability, `AmoePy` is supplied with different documentation and instruction files, i.e, a README file, a double-sided cheat sheet, and an extensive source code documentation. By developing a GUI, we facilitated the usage of `AmoePy` and potentially increased the scientific outreach of our research with respect to different target groups, i.e., biologists, biophysicists, and applied mathematicians. We described the benefits of using version control during the development process and we addressed good scientific practice guidelines by archiving the data, source codes, and instruction files appropriately. Finally, by implementing automatic tests and establishing a clear and hierarchical package structure, we improved the functionality and stability of `AmoePy`.

With the Open Traits Network (OTN), a general Open Science network across all organisms was introduced [55]. The aim of OTN is to raise awareness of Open Science principles in life sciences, to increase scientific transparency, and to accelerate the knowledge transfer by supporting the open distribution of data sets and software tools. A similar approach, but with a focus on cell migration, was taken by the Cell Migration Standardisation Organisation (CMSO). The CMSO originated from [166], in which an open data ecosystem for cell migration was introduced. This platform was then further developed and finally published in [167]. As part of CMSO, a list of standards called Minimum Information About a Cell Migration Experiment (MIACME) was published, which contains (1) general information regarding researchers, affiliations, corresponding publications, and funding, as well as (2) specific information regarding the experiment [167]. The second part is further divided with respect to metadata about: the experimental setup, the imaging conditions, and the initial/extracted data sets. Regarding the experimental setup and the imaging conditions, MIACME is based on preexisting standardization formats proposed by the two frameworks: Investigation/Study/Assay (ISA) and Open Microscopy Environment (OME), respectively. With respect to the standardization of data sets, CMSO developed a library which converts cell tracking data from different cell tracking software into a single format called `Biotracks`. Furthermore, `Biotracks` contains multiple visualization and analysis tools for cell tracking data. Currently, CMSO focuses on the standardization of one-dimensional data sets, e.g., cell tracks described by the center of mass trajectories. Hence, a standardizing format of higher dimensional data sets to describe cell migration, e.g., data containing the dynamics of two-dimensional cell contours in time and space, is yet missing. With our work, we propose a potential candidate for a format standard.

So far, many open access repositories have been established to distribute and archive research data and source codes of all disciplines. Commonly used examples for such repositories include `Zenodo`, `Dryad`, and the Open Science Framework (OSF). All of these platforms provide unique and permanent DOIs for each publication to make sharing and citation easier. Moreover, they support the usage of ORCID-numbers

(Open Researcher and Contributor ID) to clarify authorship and to enhance the visibility of each publication. Due to a built-in version control, `Zenodo` and OSF are advantageous when publishing software. In contrast, `Dryad` is intended for data publications only. However, based on a partnership with `Zenodo`, source codes can be uploaded via `Dryad` which are then published at `Zenodo`. Furthermore, `Dryad` provides a data curation, where each data set is checked for its usability. Another useful repository for our research is the Cell Image Library (CIL), a data base for cell imaging data, often published as videos or so-called Z-stack microscopy images, i.e., a time sequence of cell images. Each CIL publication is uniquely identified by a DOI and, hence, citable. Less important for our specific research work, but nonetheless noteworthy in our discipline, is `dictyBase`, an open access genome data base for *D. discoideum*.

Furthermore, many open source source packages to analyze (amoeboid) cell motility have been published. In compliance with the CMSO, an end-to-end software called `CellMissy` was introduced, containing cell segmentation and tracking routines. However, by relying on the centroid trajectory only, the analysis of contour dynamics is not yet possible in `CellMissy`. Alternatively, the image editing software `ImageJ` provides an open source plugin called `Fiji` to process imaging data [168]. In cell biology, `Fiji` is used especially for microscopy imaging data to segment and track different cells. With a `Fiji` plugin called `TrackMate-Cellpose`, one can extract two-dimensional cell contours from imaging data and track the corresponding centroid trajectory [169]. However, a method to determine the spatio-temporal mapping of cell contours is not yet provided by `TrackMate-Cellpose`. A further `Fiji` plugin to analyze pseudopods during amoeboid cell motility is called `Quimp3` and was published in [23]. In [170], `Quimp3` was extended to allow the tracking of cell contours in time and space and the analysis of the resulting contour dynamics, described as in [28]. As introduced in Chapter 4, we developed an alternative way to track and analyze contour dynamics. With `AmoePy`, this approach is now also openly accessible and, especially, easy to use.

By providing methods regarding the segmentation, analysis, and simulation of contour dynamics, `AmoePy` is developed as versatile all-in-one package. For this reason, it can be potentially used for many different applications in biology and biophysics. Thus, we envision `AmoePy` to be of greater value for the cell migration community. More generally, `AmoePy` has the potential to be a role model for future Open Science contributions. With our approach, we have demonstrated measures to make scientific work more transparent and reproducible and we have outlined the great benefits, personally and collectively, of following Open Science principles. For the future, we aim to publish `AmoePy` on `GitHub`, to make suggestions and source code contributions by the community possible. Moreover, we would like to provide `AmoePy` directly via the `Conda` package manager to further increase its usability and public outreach.

Outlook

7

In this thesis, we proposed a systematic and data-driven approach to analyze and simulate amoeboid cell motility. This approach consists of (1) a mathematical framework based on a one-parameter family of regularized flows to describe the dynamics of 2D cell contours and (2) a three-component contour dynamics model to analyze experimental contour data by inferring key motility properties and to simulate realistic and qualitatively different contour dynamics. As part of an Open Science approach, both methods were implemented and supplied in an easy-to-use way within our software package `AmoePy`. All three topics were presented and discussed in separate parts of this thesis, see Chapters 4–6.

In this chapter, we want to broaden the discussion beyond the results of this thesis and provide general perspectives for further research. Additional to contour data based on *D. discoideum*, we applied the mathematical framework from Chapter 4 to other organisms, i.e., early embryonic killifish cells (*Fundulus heteroclitus*) and keratocytes cultured from Central American cichlids (*Hypsophrys nicaraguensis*), and for different experimental setups, i.e., bright-field microscopy, fluorescence microscopy, and varying frame rates. Other potential fields to apply our method include different locomotion types, e.g., fan-shaped locomotion [75, DS4], amoeboid swimmers [134, 135, 171, 172], mesenchymal cell migration [5, 172, 173], and collective/multicellular cell migration [5, 153]. In [DS4], we applied the above method to analyze mutated *D. discoideum* cells, where we identified reversible motility switches between the amoeboid and the fan-shaped type. Our analysis tools could be also used to study amoeboid migrating cells, e.g., lymphocytes and neutrophils as well as lymphoma, leukemia, and other cancer cells, for a better understanding of physiological/medical processes such as wound healing, the immune system response, and the invasion and metastasis of cancer [5]. In Chapter 4, we pointed out that previous approaches, as well as ours, to connect consecutive cell contours in time and space are not intrinsically defined and that they are bound to different assumptions [24, 28, 29]. For this reason, the development of a contour mapping method, which is physiologically and data-driven based, would be an interesting, but experimentally challenging, objective.

As part of the three-component contour dynamics model from Chapter 5, the persistence of a simulated cell track is based on a polarization parameter and is fixed for the entire time interval. Hence, a more detailed incorporation of chemotaxis into the model is possible, e.g., by adding a fourth component to the model, which is based on a time-dependent vector field representing chemotactic stimuli. We applied the model to experimental contour data to infer the underlying protrusion component, which was then compared to commonly used biomarkers: the F-actin density close to the membrane and the local motion. While this was only a first step, we envision our model to be used for the investigation of other intracellular mechanisms, e.g., the signaling pathways of amoeboid migrating cells during chemotaxis [174]. Due to the low model complexity, our model can be simply applied to other organisms and varying experimental setups. Other applications include, for example, the transition from single-cell migration to collective cell migration [5] and the detection of motility mode switches, e.g., amoeboid to fan-shaped [DS4] or mesenchymal to amoeboid [5, 172]. Another interesting research question is the parameter estimation/tuning of the underlying Hawkes process in our model such that the predicted motility patterns correspond to experimental observations as much as possible.

Regarding our software package `AmoePy`, introduced in Chapter 6, we aim to further increase its usability and outreach. In this context, we consider its publication on `GitHub` to enable other researchers to make suggestions and source code contributions. In a next step, we would like to facilitate the installation process of `AmoePy`, by deploying it directly via the `Conda` package manager. Alternatively, to the current graphical user interface, a browser-based interface could improve the usability of `AmoePy` even more.

Conclusion. In this thesis, we developed a mathematical framework and a computational model, which allow a systematic, quantitative, and data-driven analysis of contour dynamics underlying amoeboid cell motility. By means of regularized flows, we defined a spatio-temporal coordinate system of evolving cell contours. Applied to experimental contour data, primarily recorded for *D. discoideum*, this framework allows a rigorous analysis of different cell tracks, where the main protrusion and retraction events are identified and classified in a fully automated way. This framework was then used as part of a novel contour dynamics model, which enables us to (1) simulate qualitatively different and realistic contour dynamics and (2) analyze experimental contour data by inferring key motility characteristics regarding protrusions and retractions. Due to its intuitively comprehensible approach and the low model complexity, we deem the model to be suitable for a wide range of different organisms, experimental setups, and physiological/biomedical applications. By providing the above methods as part of an easy-to-use open source software package, this work is of greater value, in particular, for the cell migration community and, in general, for researchers of different fields, e.g., biostatistics, biophysics, and medicine. Finally, by following Open Science principles, this work enables other researchers to reproduce our scientific findings and represents a valuable contribution to the Open Science movement.

Appendix related to Chapter 4 A

A.1 Supporting information: regularized contour flow method

In this section, we present further details of the regularized contour flow method presented as in Chapter 4. First, we provide information regarding the implementation of the regularized contour flow method (RCFM) within `AmoePy`, our Python-based toolbox for analyzing and simulating amoeboid cell motility [DS3]. Subsequently, we describe the gradient descent method which is used to solve the minimization problem from Eq (4.23). More precisely, we show that predictions of the functional gradients are provided by the GPR as analytical by-products. This will reduce the number of iterations of the gradient descent method and, therefore, reduce the computational cost of the algorithm substantially. Furthermore, we present test cases which are used to validate the RCFM. These test cases as well as multiple experimental cell tracks are fully accessible in our `AmoePy` software package.

Objective function. Note that the optimal flow is defined as the flow that solves the following minimization problem:

$$\phi_{k,\lambda} = \underset{\phi_k}{\operatorname{argmin}} \; F_k[\phi_k] + \lambda U_k[\phi_k], \quad \lambda > 0, \tag{A.1}$$

where the two functionals are given by:

$$F_k[\phi_k] \simeq F_k\left[\theta_{k+1}|\theta_k\right] = \frac{1}{N\delta t^2} \sum_{i=0}^{N-1} \left\|\Phi_{k+1}(\theta_{k+1,i}) - \Phi_k(\theta_{k,i})\right\|^2, \tag{A.2}$$

$$U_k[\phi_k] \simeq U_k\left[\theta_{k+1}|\theta_k\right] = N \sum_{i=0}^{N-1} \left|\theta_{k+1,i+1} - \theta_{k+1,i}\right|^2, \tag{A.3}$$

with $\theta_k, \; \theta_{k+1} \in [0, 2\pi)^N$ denoting the normalized arc length coordinates for N segmentation points at time t_k and t_{k+1}. For two consecutive parametrizations θ_{k+1} and θ_k and regularization parameter $\lambda \in \mathbb{R}^+$, the objective function $H_k[\phi_k] := F_k[\phi_k] + \lambda U_k[\phi_k]$ is given by

$$\begin{aligned} H_k[\phi_k] \simeq H_k\left[\theta_{k+1}|\theta_k\right] = \frac{1}{N\delta t^2} &\sum_{i=0}^{N-1} \left\|\Phi_{k+1}(\theta_{k+1,i}) - \Phi_k(\theta_{k,i})\right\|^2 \\ &+ \lambda N \sum_{i=0}^{N-1} \left|\theta_{k+1,i+1} - \theta_{k+1,i}\right|^2. \end{aligned} \tag{A.4}$$

From the following optimization problem

$$\theta_{k+1} := \underset{\tilde{\theta}_{k+1} \in [0,2\pi)^M}{\operatorname{argmin}} \quad H_k\left[\tilde{\theta}_{k+1} | \theta_k\right], \tag{A.5}$$

we obtain the arc length parametrization of the next contour.

Derivatives of objective function. In order to solve the optimization problem, we used built-in gradient descent methods from the Python package `SciPy`. In this context, we have chosen `'L-BFGS-B'` as minimizer due to its short computation time and the usage of bound constraints. Alternatively, `'trust-constr'` offers a constrained minimization with a slightly slower computation times. For more details about both minimizers, see [175, 176] and [177, 178]. The Jacobian and Hessian of H_k, which is required for some custom minimizers, are given by the following formulas:

$$\begin{aligned}
\frac{\partial H_k}{\partial \theta_{k+1,i}} &= 2a \cdot \left\langle \Phi_{k+1}\left(\theta_{k+1,i}\right) - \Phi_k\left(\theta_{k,i}\right), \frac{\partial \Phi_{k+1}}{\partial \theta_{k+1,i}} \right\rangle \\
&\qquad - 2b \cdot \left(\theta_{k+1,i-1} + \theta_{k+1,i+1}\right) + 4b \cdot \theta_{k+1,i} \\
\frac{\partial^2 H_k}{\partial \theta_{k+1,i}^2} &= 2a \cdot \left\langle \Phi_{k+1}\left(\theta_{k+1,i}\right) - \Phi_k\left(\theta_{k,i}\right), \frac{\partial^2 \Phi_{k+1}}{\partial \theta_{k+1,i}^2} \right\rangle \\
&\qquad 2a \cdot \left\| \frac{\partial \Phi_{k+1}}{\partial \theta_{k+1,i}} \right\|^2 + 4b, \\
\frac{\partial^2 H_k}{\partial \theta_{k+1,i}\, \partial \theta_{k+1,j}} &= \begin{cases} -2b & |i-j| = 1, \\ 0 & 1 < |i-j| < N-1, \\ -2b & |i-j| = N-1, \end{cases}
\end{aligned}$$

where $a := 1/(N\delta t^2)$ and $b := \lambda N$ and $i, j \in \{0, \ldots, N-1\}$. Conveniently, the derivatives $\partial\Phi_{k+1}/\partial\theta_{k+1,i}$ and $\partial^2\Phi_{k+1}/\partial\theta^2_{k+1,i}$ are a by-product from the Gaussian process regression obtained by Eq (3.15) and (3.16).

Linear constraints. In order to avoid mapping violations, one can choose a gradient descent method with following constraints

$$\begin{aligned}
I: &\quad 0 \leq \theta_{k,j+1} - \theta_{k,j} \leq 2\pi \qquad \text{and} \\
II: &\quad 0 \leq \theta_{k,N-1} - \theta_{k,0} \leq 2\pi.
\end{aligned}$$

Both inequalities can be summarized by the following linear constraint:

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ -2\pi \end{pmatrix} \leq \begin{pmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 1 & & & -1 \end{pmatrix} \cdot \begin{pmatrix} \theta_{k,0} \\ \theta_{k,1} \\ \vdots \\ \theta_{k,N-1} \end{pmatrix} \leq \begin{pmatrix} 2\pi \\ \vdots \\ 2\pi \\ 0 \end{pmatrix}.$$

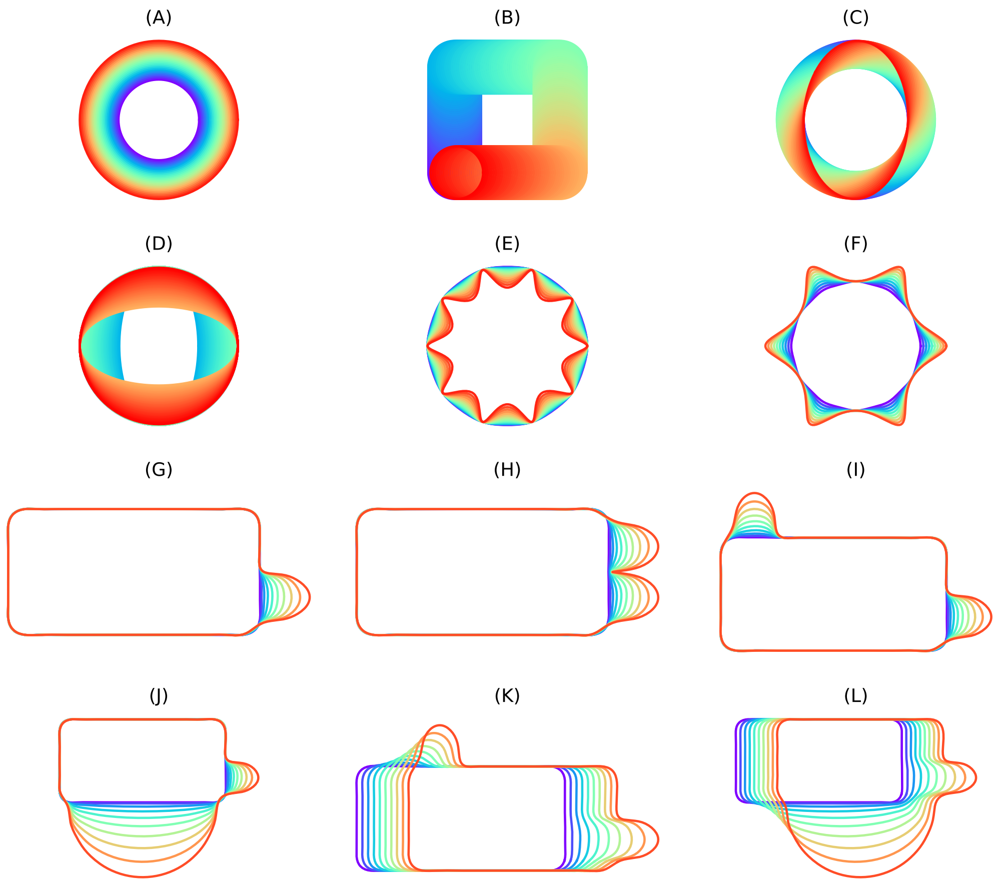


Figure A.1. **Selection of test cases accessible in** `AmoePy`: Pulsating circle **(A)**, translation of circle **(B)**, rotating ellipse **(C)**, circle to ellipse transformation **(D)**, inward membrane changes **(E)**, outward membrane changes **(F)**, single protrusion **(G)**, neighboring protrusions **(H)**, distant protrusions **(I)**, different sized protrusions and different protrusions during translation **(K, L)**. The time period of each excerpt is color-coded from purple to red.

Since the second functional in Eq (A.1) already penalizes mapping violations, i.e., $\theta_{k,i+1} - \theta_{k,i} \leq 0$ for $i \in \{0, \ldots, N-1\}$, we have chosen a gradient descent minimizer, which skips the check of linear constraint violations is skipped for the sake of a faster computation time.

Method validation on test cases. Our software package `AmoePy` provides several simple contour dynamics as displayed in Fig A.1. These test cases consist of basic geometric transformations (A-F) and different protrusion patterns (G-L). Many tracks are periodic to further challenge the algorithm. For periodic contour dynamics, only the first period is displayed in Fig A.1, while the kymographs in File A.1 provide information for all periods. The starting contour is highlighted in purple, whereas the last contour is highlighted in red. The exact time intervals of each track can be extracted from File A.1.

Special features of the local dispersion are nicely illustrates in these kymographs. In the example case of the pulsating circle (A), the local dispersion is equal to zero for the entire contour and time interval, since the local dispersion was defined as a concentration/stretching rate of virtual markers on the unit circle $\mathcal{S}^1$. Moreover, cases (G-L) illustrate a non-local feature of the marker dispersion: Strong protrusions at one part of the contour may also affect the dispersion of markers at the rest of the contour. In `AmoePy`, a detailed documentation is included which can be used to recreate the corresponding kymographs as well as animated videos of these test cases.

A.2 Supporting computations: Cost functionals

In this section, we provide the step-by-step derivations for some of the statements presented in the Section 4.2.

(I) Cost functional $U_k[\phi_k]$. Recall that virtual markers along a contour Γ_k are distributed according to a density $\mu_k(\theta)$. The density of virtual markers on the consecutive contour induced by ϕ_k can be described with

$$\mu_{k+1}(\phi_k(\theta))d\theta = \frac{\mu_k(\theta)d\theta}{\partial_\theta \phi_k(\theta)}, \tag{A.6}$$

where the underlying flow is defined by

$$\theta_{k+1} = \phi_k(\theta_k).$$

The second functional, which quantifies the degree of non-uniformity, is defined by

$$U[\mu] := \int_0^{2\pi} \frac{\mathrm{d}\theta}{\mu(\theta)}. \tag{A.7}$$

By applying the substitution rule for integrals to $\mu_{k+1}(\phi_k(\theta))$, we obtain the following expression

$$U_k[\phi_k] = \int_0^{2\pi} \frac{\partial_\theta \phi_k(\theta)}{\mu_{k+1}(\phi_k(\theta))}\mathrm{d}\theta \overset{(A.6)}{=} \int_0^{2\pi} \frac{\partial_\theta \phi_k(\theta)^2}{\mu_k(\theta)}\mathrm{d}\theta. \tag{A.8}$$

(II) Cost functional $U_k[\phi_k]$. Note that the mapping from Γ_0 to Γ_k is denoted as $\chi_k(\theta)$ and defined by

$$\chi_{k+1}(\theta) = \phi_k(\chi_k(\theta)), \quad \chi_0(\theta_0) = \theta_0.$$

Furthermore, the density of virtual markers along the cell contour Γ_k is defined by

$$\mu_k(\chi_k(\theta_0)) = \frac{1}{2\pi \cdot \partial_{\theta_0}\chi_k(\theta_0)}. \tag{A.9}$$

Then, we obtain the following expression

$$\begin{aligned} U_k[\mu] &:= \int_0^{2\pi} \frac{\mathrm{d}\theta}{\mu(\theta)} \\ &= \int_0^{2\pi} \frac{1}{\mu_k\left(\chi_k(\theta_0)\right)} \cdot \partial_{\theta_0}\chi_k(\theta_0)\mathrm{d}\theta_0 \end{aligned}$$

$$\overset{(A.9)}{=} 2\pi \int_0^{2\pi} |\partial_{\theta_0}\chi_k(\theta_0)|^2 \,\mathrm{d}\theta_0.$$

From there on, we can deduce the formula from Eq (4.17):

$$\begin{aligned} U_k[\phi_k] &= 2\pi \int_0^{2\pi} |\partial_{\theta_0}\phi_k\left(\chi_k(\theta_0)\right)|^2 \,\mathrm{d}\theta_0 \\ &= 2\pi \int_0^{2\pi} |\partial_{\theta_0}\chi_{k+1}(\theta_0)|^2 \,\mathrm{d}\theta_0. \end{aligned}$$

(III) Cost functional $F_k[\phi_k]$. Of note, the square mean velocity of the flow is defined by

$$F_k[\phi_k] = \int_0^{2\pi} \|V_k(\theta)\|^2 \mu_k(\theta)\mathrm{d}\theta,$$

and the translation vectors Φ, V, and W are linked by the following formula

$$\frac{\Phi_{k+1}(\phi_k(\chi_k(\theta_0))) - \Phi_k(\chi_k(\theta_0))}{\delta t} = V_k(\chi_k(\theta_0)) = W_k(\theta_0). \tag{A.10}$$

Again, by using the substitution formula for integrals, we obtain the following expression of the cost functional

$$\begin{aligned} F_k[\phi_k] &:= \int_0^{2\pi} \|V_k(\theta)\|^2 \mu_k(\theta)\mathrm{d}\theta \\ &= \int_0^{2\pi} \|V_k(\chi_k(\theta_0))\|^2 \mu_k(\chi_k(\theta_0)) \cdot \partial_{\theta_0}\chi_k(\theta_0)\mathrm{d}\theta_0 \\ &\overset{(A.9)}{=} \frac{1}{2\pi}\int_0^{2\pi} \|V_k(\chi_k(\theta_0))\|^2 \,\mathrm{d}\theta_0 \\ &\overset{(A.10)}{=} \frac{1}{2\pi}\int_0^{2\pi} \left\|\frac{\Phi_{k+1}(\phi_k(\chi_k(\theta_0))) - \Phi_k(\chi_k(\theta_0))}{\delta t}\right\|^2 \mathrm{d}\theta_0. \end{aligned}$$

A.3 Supporting figures

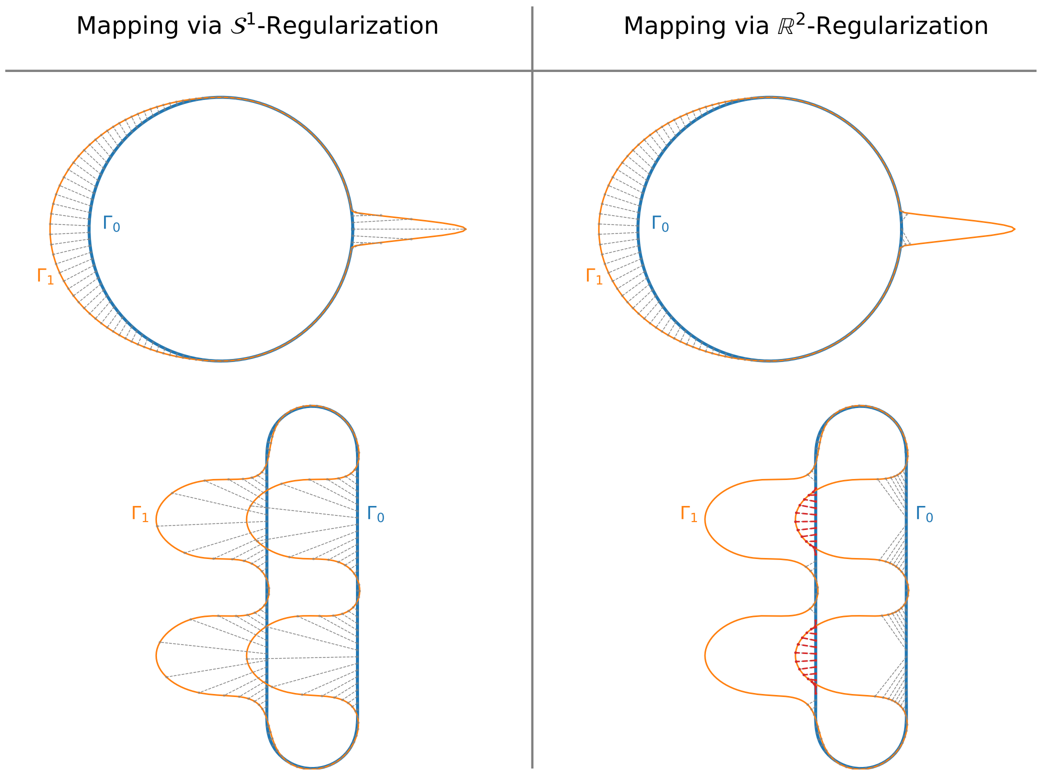

Mapping via $\mathcal{S}^1$-Regularization
Mapping via $\mathbb{R}^2$-Regularization
Γ_0
Γ_1
Γ_0
Γ_1
Γ_1
Γ_0
Γ_1
Γ_0

Figure A.2. Comparison of virtual marker mappings obtained with different regularization schemes. On the left hand side, $\mathcal{S}^1$-regularization was used in which neighboring virtual markers are regularized w.r.t their normalized arc length coordinates. On the right hand side, the $\mathbb{R}^2$-distances of neighboring virtual markers is used for regularization. While the $\mathbb{R}^2$-regularization fails during large shape deformations, producing wide gaps between virtual markers and topological mapping violations (red lines), satisfying contour mappings are provided by the $\mathcal{S}^1$-regularization as used by our method.

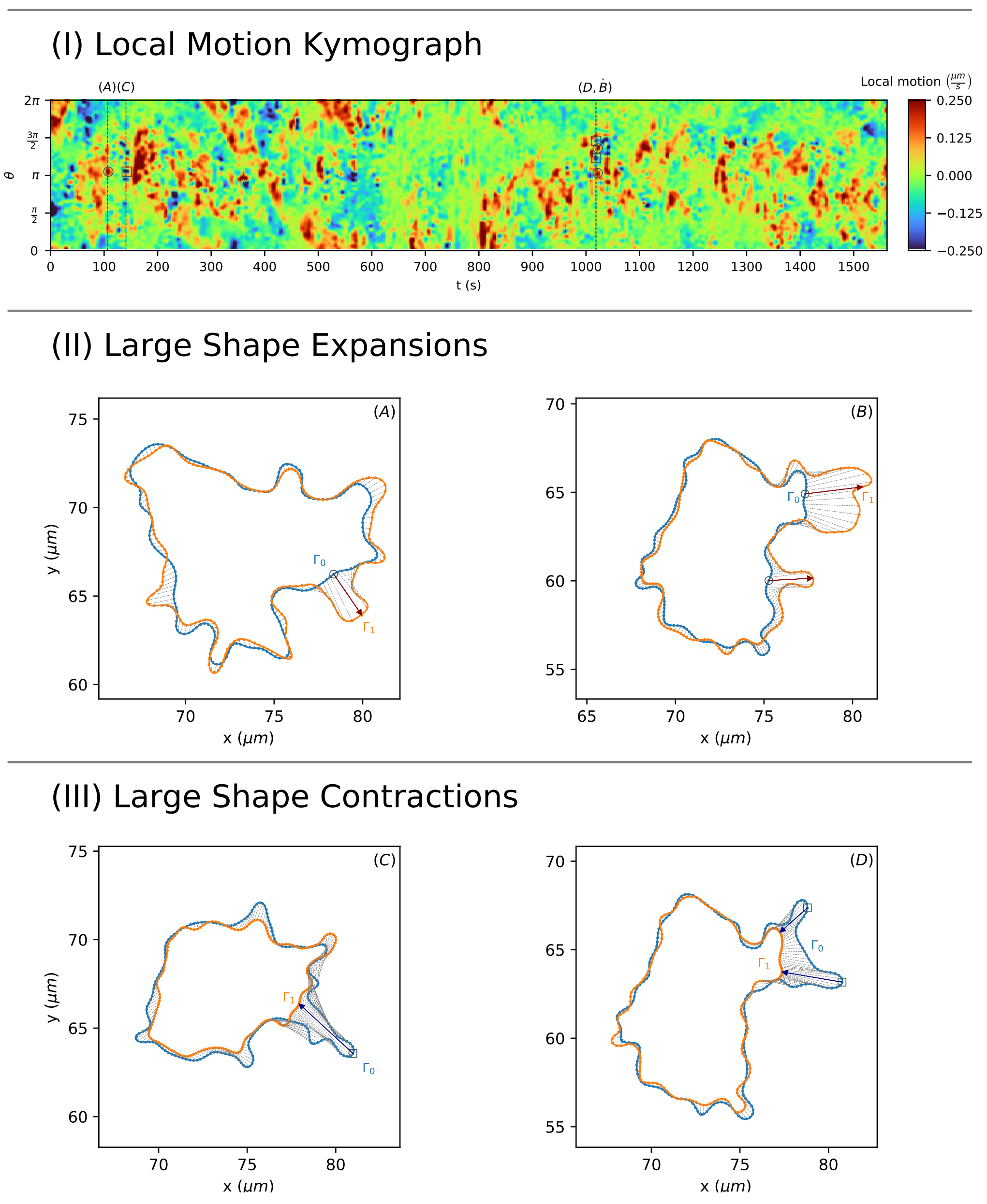


Figure A.3. Cell track with large shape deformations. As in previous kymographs, a strongly regularized (global) flow is used to define the underlying coordinate system. Then, based on a weakly regularized (local) flow ($\lambda = 0.01$), the local motion is computed (first row). Compared to previous cell tracks, the imaging frequency of this cell track is roughly three times lower ($\delta t \approx 3.13s$), which results in larger shape deformations. Examples showing the local flow between two consecutive frames are displayed for expansions (second row) and contractions (third row). Expansion and contraction events, indicated by red and blue arrows, respectively, are also displayed in the local motion kymograph (gray circles/squares, respectively).

```
Algorithm 3: MinimizeFunctional
Input: $\gamma_{\text{start}}$, $\gamma_{\text{end}}$, $r \in (0,1)$, $\sigma_n > 0$, $\lambda \geq 0$
Output: $\boldsymbol{\theta}_{\text{eval}}$

/* Compute normalized arc length coordinates                          */
$\boldsymbol{\theta}_{\text{start}}$ = NormalizedArcLengthCoordinates($\gamma_{\text{start}}$);
$\boldsymbol{\theta}_{\text{end}}$ = NormalizedArcLengthCoordinates($\gamma_{\text{end}}$);

/* Setting initial value                                              */
$\boldsymbol{\theta}_{\text{eval}} = \boldsymbol{\theta}_{\text{start}}$;

/* Perform gradient descent to minimize Functional                    */
/* The gradient is a by-product of GPR                                */
/* See Supporting Information (S1 File) for more details              */
find $\theta_{\text{eval}}$ which minimizes $f_{\text{obj}}$
    $\Gamma_{\text{flow}}$ = GPR($\gamma_{\text{end}}$, $\boldsymbol{\theta}_{\text{end}}$, $\boldsymbol{\theta}_{\text{eval}}$, $r$, $\sigma_n$);
    $f_{\text{obj}}$ = Functional($\gamma_{\text{start}}$, $\Gamma_{\text{flow}}$, $\boldsymbol{\theta}_{start}$, $\boldsymbol{\theta}_{eval}$);
end

return $\boldsymbol{\theta}_{eval}$
```

Figure A.4. Schematic overview of an algorithm to obtain coordinate markers from solving the optimization problem as in Eq (4.25).

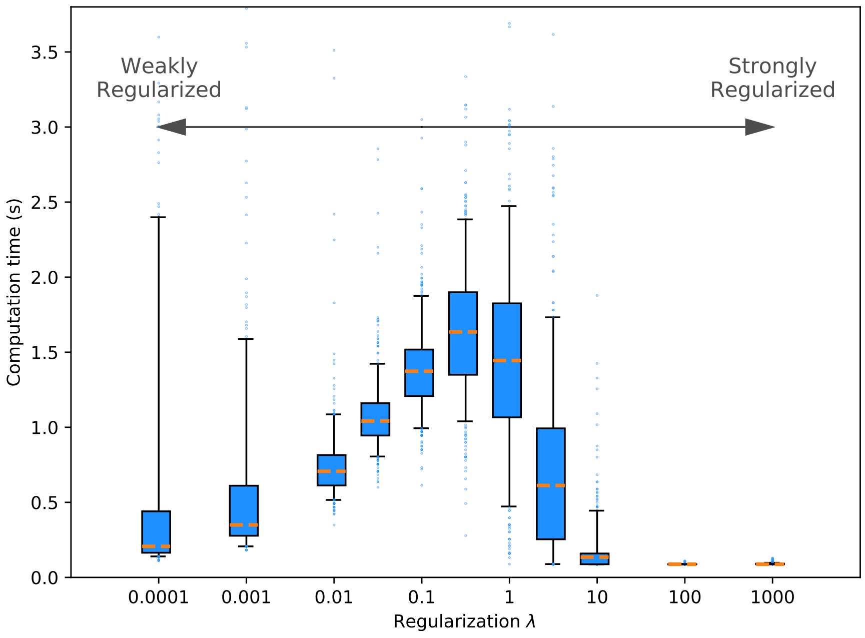


Figure A.5. Computation times of the algorithm 'Regflow' from Fig 4.2 for different values of the regularization parameter λ for the mapping between two consecutive contours. The statistic is taken over 500 pairs of consecutive contours of the cell track displayed in Fig 4, each based on 400 virtual markers. For each λ, the median (orange lines), the upper and lower quartile (blue boxes), and the 5th and 95th percentile (black lines) are shown.

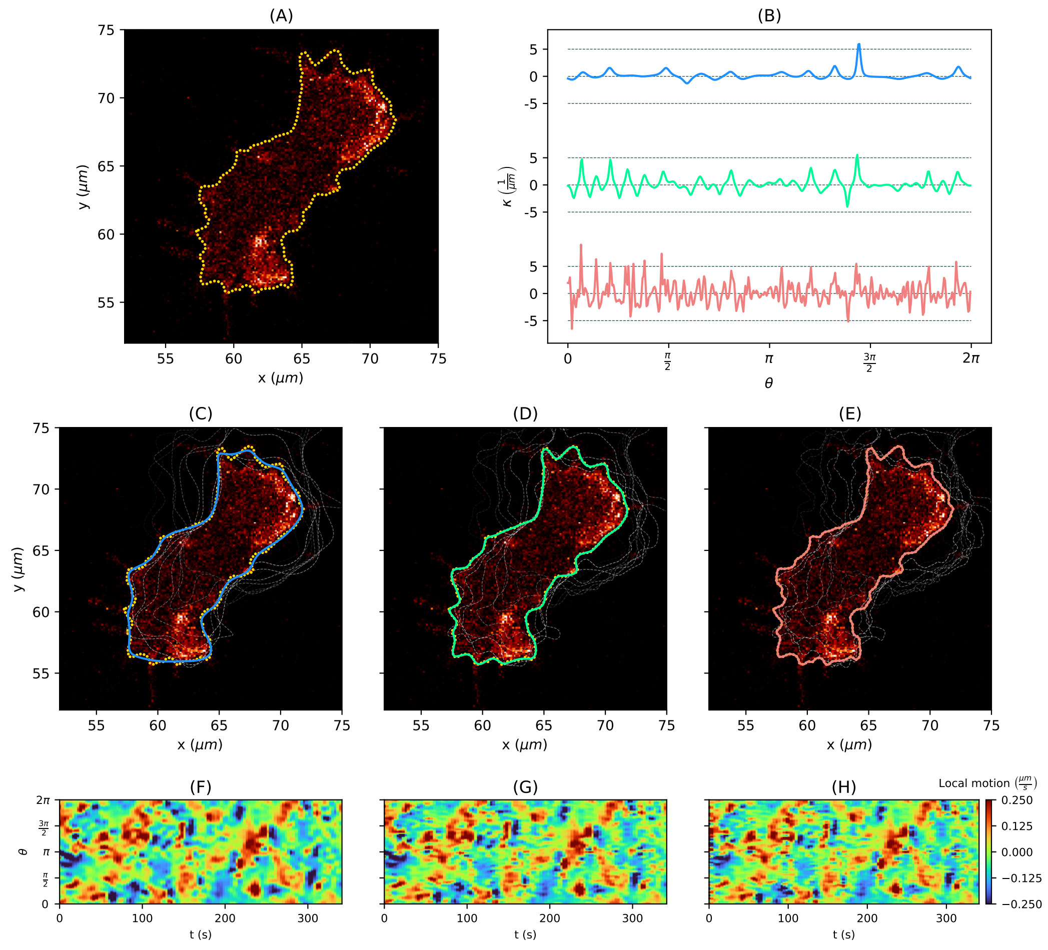


Figure A.6. **Cell contour based on noisy fluorescence images** estimated via GPR for different hyperparameters and corresponding curvature. Discrete set of points (yellow) segmented from a noisy fluorescence image **(A)**. By using Gaussian process regression with varying parameters of the underlying Poisson kernel, different estimations of the membrane can be obtained. The curvature of these estimated contours are shown in **(B)**. The parameters were chosen as follows: $r = 0.4$ (blue), $r = 0.65$ (green), and $r = 0.9$ (red). In (**C**, $r = 0.4$), the resulting contour is highly underfitted which leads to a strongly regularized curvature. In (**E**, $r = 0.9$), an overfitting effect can be observed, resulting in high fluctuations of the curvature. In (**D**, $r = 0.65$), a more plausible parameter was chosen leading to an accurate approximation while preserving the main characteristics of the curvature. The corresponding local motion kymographs are shown in panel **(F-H)**. For (**F**, $r = 0.4$), some details are not resolved. In contrast, only few differences can be observed between (**G**, $r = 0.65$) and (**H**, $r = 0.9$), which shows that the estimate for $r = 0.65$ is already an accurate approximation of the contour.

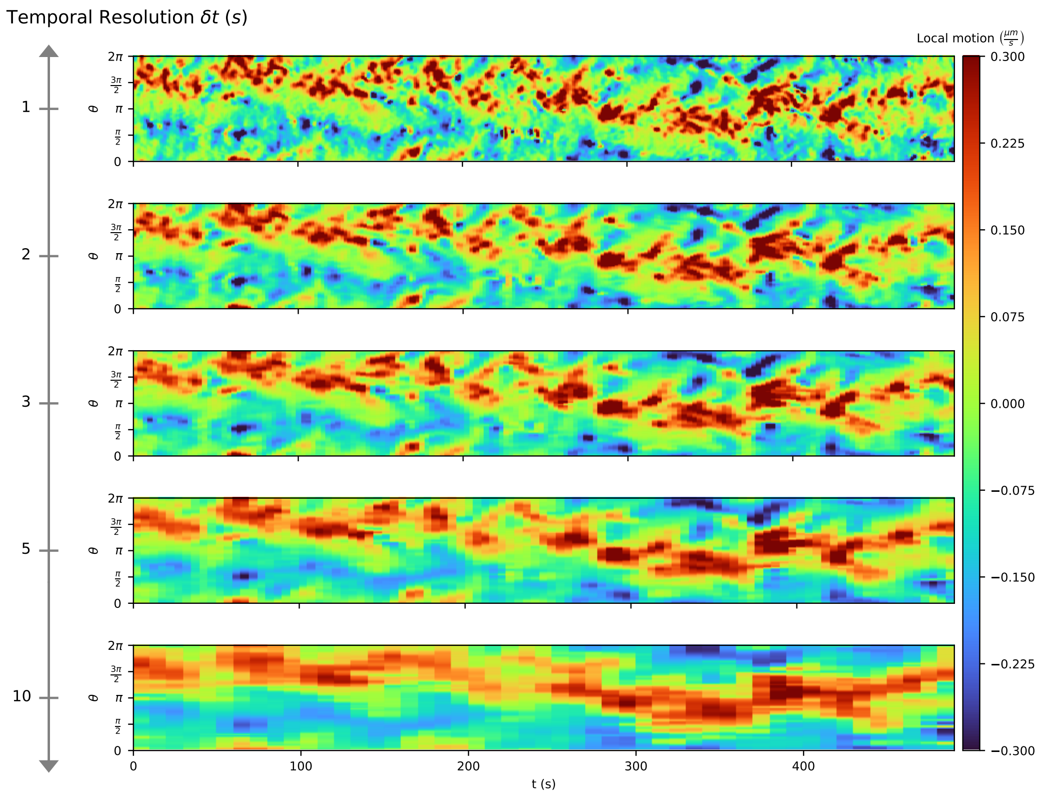


Figure A.7. **Comparison of local motion kymographs obtained with different imaging frequencies:** $\delta t \in \{1, 2, 3, 5, 10\}$. At the top, the kymograph is based on one image/contour per second. In the kymographs below, the underlying contour flows were computed for every 2nd, 3rd, 5th, and 10th image. For decreasing imaging frequencies, the identification of local membrane changes becomes more difficult. However, the algorithm is stable even for a lower temporal resolution ($\delta t > 3s$) producing contour flows without mapping violations while capturing global features of the contour dynamics.

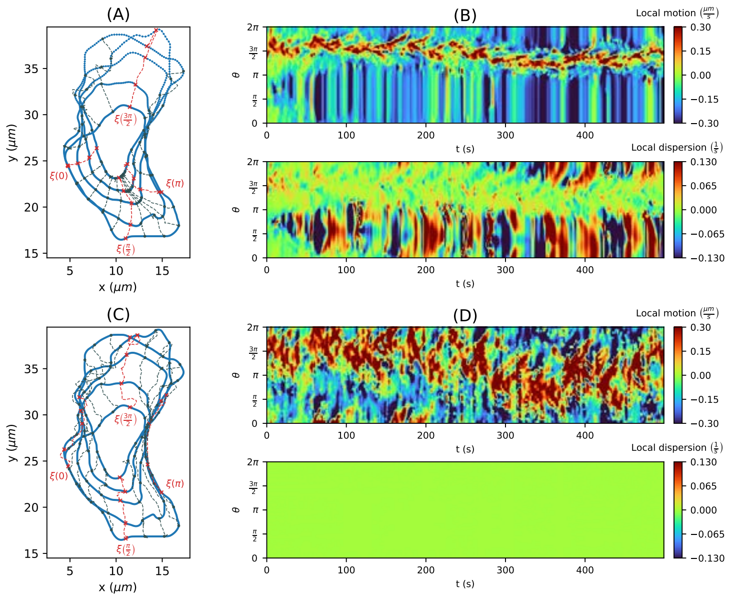


Figure A.8. Comparison of global flows in cases of no, weak and strong regularization. Weakly regularized global flow ($\mathbf{A}$, $\lambda = 1$) with corresponding kymographs: Local motion (top) and local dispersion (bottom). Clustering and thinning effects appear at the back side and front side of the cell, respectively, which results in an overproportional display of a short retractive contour segment. For the case of the strongly regularized flow ($\mathbf{C}$, $\lambda = 1000$) we observe a more uniform distribution of virtual markers along the cell. Therefore, the corresponding local dispersion kymograph becomes less informative ($\mathbf{D}$, bottom).

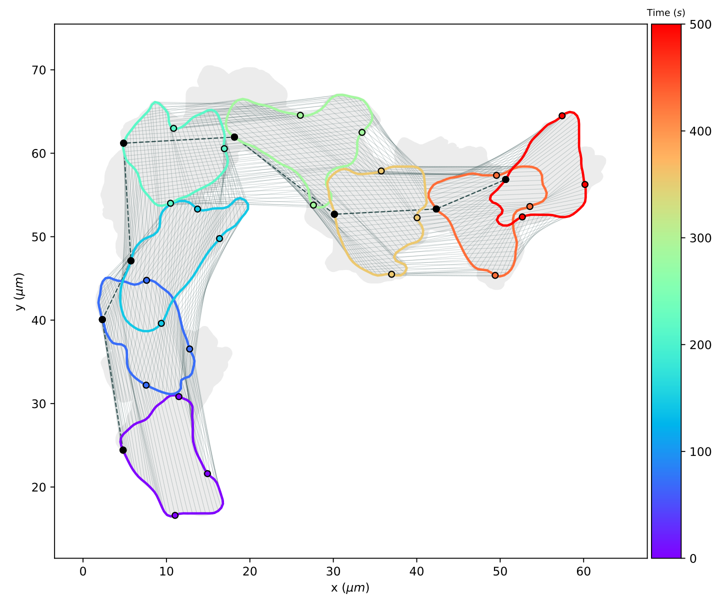


Figure A.9. Extreme test scenario to further challenge our algorithm. In this test case, only 8 out of 500 contours are taken into account. The underlying coordinate system is defined by a strongly regularized flow ($\lambda = 1000$) and is depicted as gray lines. For illustration, we have marked four points on the contour with normalized arc length 0, $\frac{\pi}{2}$, π, $\frac{3\pi}{2}$ and highlighted the mapping of one of them. Additionally, the initial cell track is highlighted as gray area. Noticeable, no mapping violations were produced. Moreover, the strongly regularized flow prevents clustering and thinning effects of virtual markers.

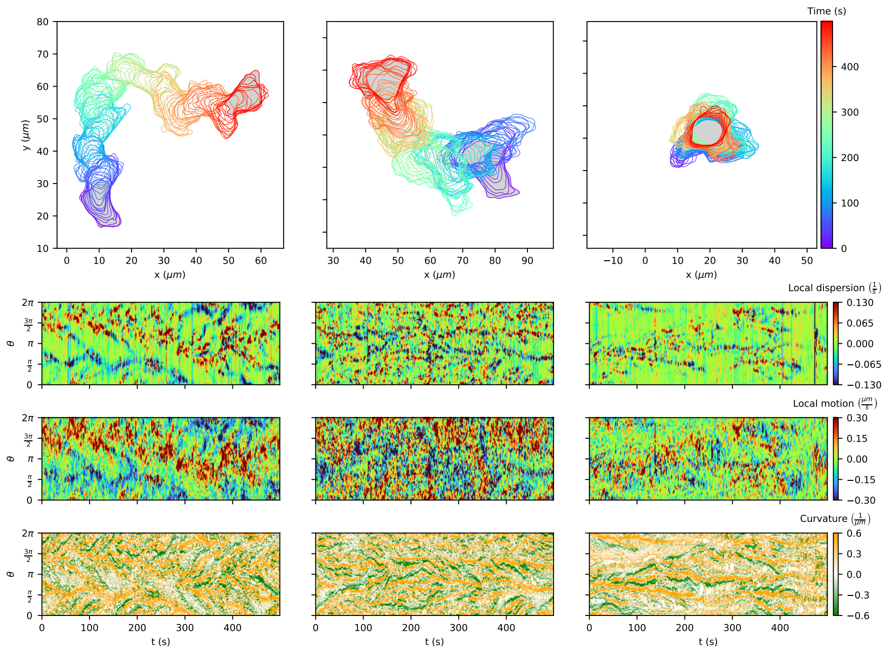


Figure A.10. Kymographs as in Fig 4.6 without prior smoothing.

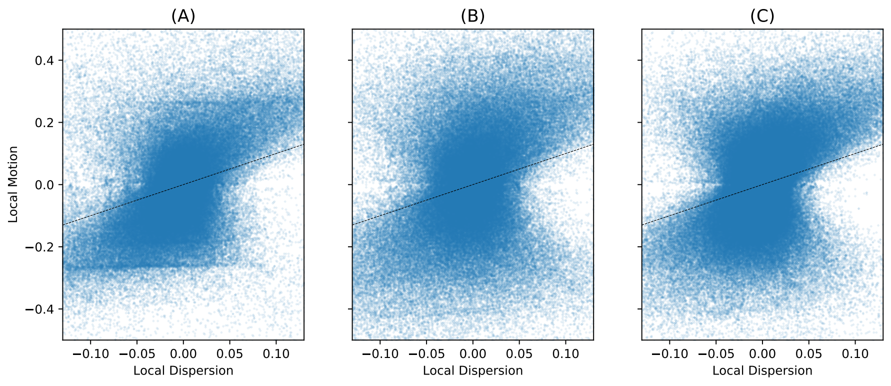


Fig S12. Correlation between local dispersion and local motion. The correlation is shown for different example cell tracks: Persistently motile **(A)**, medium motile **(B)** and rather stationary cell **(C).**

Figure A.11. Correlation between local dispersion and local motion. The correlation is shown for different example cell tracks: Persistently motile **(A)**, medium motile **(B)** and rather stationary cell **(C)**.

Expansion collection, threshold: 0.087

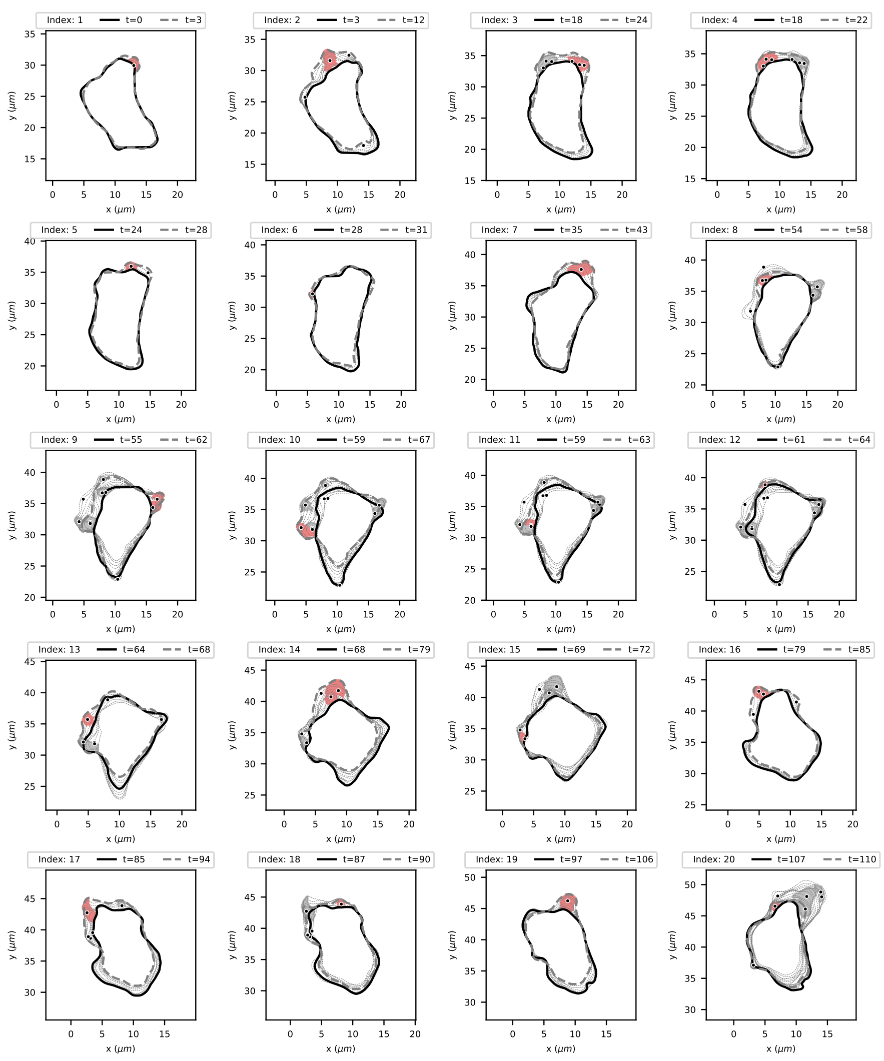


Figure A.12. Collection of identified expanding areas and events of high intensity for the persistently motile cell. Only features with minimal persistence length $\Delta t \geq 3$ are shown. For a complete collection of all identified contour expansions, see File A.4.

A.4 Supporting files

File A.1. Collection of 13 artificial cell tracks with corresponding kymographs showing local dispersion, local motion, and curvature. In the figure at the top of each page, the period of time of several cell tracks was shortened for illustrative purposes due to overlapping contours. URL: https://doi.org/10.1371/journal.pcbi.1009268.s005 (PDF)

File A.2. Collection of 12 cells with high and medium motility and different grades of persistence. For each track the kymographs of local dispersion, local motion and curvature are shown, followed by plots as in Fig 4.9 and 4.10. The cells are sorted in descending order regarding the area of the entire contour track. URL: https://doi.org/10.1371/journal.pcbi.1009268.s014 (PDF)

File A.3. Collection of 12 stationary cells. The file is structured as in File A.2. URL: https://doi.org/10.1371/journal.pcbi.1009268.s015 (PDF)

File A.4. Collection of identified expanding areas and events of high intensity for the persistently motile cell. Only features with minimal persistence length $\Delta t \geq 3$ are shown. URL: https://doi.org/10.1371/journal.pcbi.1009268.s016 (PDF)

A.5 Supporting videos

Video A.1. Persistently motile cell at tenfold speed, see S1 Video of [DS1]. URL: https://doi.org/10.1371/journal.pcbi.1009268.s017 (MP4)

Video A.2. Weakly motile cell at tenfold speed, see S2 Video of [DS1]. URL: https://doi.org/10.1371/journal.pcbi.1009268.s018 (MP4)

Video A.3. Stationary cell at tenfold speed, see S3 Video of [DS1]. URL: https://doi.org/10.1371/journal.pcbi.1009268.s019 (MP4)

Appendix related to Chapter 5 B

B.1 Hawkes process: Expected number of offspring

As described in Chapter 5, the underlying kernel of the Hawkes process is given by $g(t,\theta) = g_1(t) \cdot g_2(\theta)$, separated into a temporal component $g_1(t)$ and a spatial component $g_2(\theta)$. In this context, the temporal kernel is given by

$$g_1(t) = \alpha\beta\, t\, e^{-\beta t},$$

with arrival intensity $\alpha > 0$ and exponential decay rate $\beta > 0$. Furthermore, the spatial kernel is defined as the von Mises distribution

$$g_2(\theta) = \frac{e^{\kappa_M \cos(\theta)}}{2\pi I_0(\kappa_M)},$$

with $\kappa_M > 0$ as concentration parameter and $I_0(\kappa_M)$ denoting the modified Bessel function of order 0.

The expected number of offspring under this Hawkes process is then given by the following integral

$$m = \int_0^{2\pi} \int_0^{\infty} g(t,\theta)\; dt\; d\theta.$$

Since g_1 and g_2 are continuous and, therefore, g as well, we can apply Fubini's theorem. Furthermore, due to g_2 being a probability density function, we obtain

$$\begin{aligned} m &= \int_0^{2\pi} g_2(\theta)\; d\theta \cdot \int_0^{\infty} g_1(t)\; dt \\ &= 1 \cdot \int_0^{\infty} \alpha\beta\, t\, e^{-\beta t}\; dt \\ &= \left[-\frac{\alpha e^{-\beta t}\,(\beta t + 1))}{\beta} \right]_0^{\infty} \\ &= \frac{\alpha}{\beta} \end{aligned}$$

For the parameters $\alpha = 0.5s^{-1}$ and $\beta = 0.4s^{-1}$ used in our paper, we obtain the expected number of offspring $m = 0.8$. Given a background intensity $\lambda_0 = 1s^{-1}$ and a time span $T = 500s$, we can compute the expected number of events

$$\lambda_0 \cdot T \cdot \sum_{k=0}^{\infty} m^k = 2500.$$

Therefore, to obtain the same amount of events with a Poisson process, i.e., with one generation of events only, the background intensity must be set to $\lambda_0 = 5s^{-1}$.

B.2 Estimation of model weights

In this section, we describe our approach how to estimate the model weights w_{prot}, w_{APCSF}, and w_{AAF} when replicating experimental cell tracks with our model.

In this context, we will make use of the local motion kymograph derived from the experimental contour dynamics; see [DS1] for more details. More precisely, we tune f_{APCSF} and f_{AAF} such that negative regions (i.e. retractions) of the local motion are mainly captured by the above components. For this reason, we introduce the residuals $r_{k,i}$ defined by

$$\begin{aligned} r_{k,i} &= \text{LM}_{k,i} - f_{\text{APCSF}}\left(t_k, \theta_i\right) - f_{\text{AAF}}\left(t_k, \theta_i\right) \\ &\approx f_{\text{prot}}(t_k, \theta_i) \end{aligned}$$

with local motion $\text{LM}_{k,i}$ at time t_k and virtual marker θ_i.

First, we propose the following two sums of squared residuals

$$\begin{aligned} S &:= \sum_{k,i\colon \text{LM}_{k,i}<0} r_{k,i}^2, \\ S^+ &:= \sum_{k,i\colon \text{LM}_{k,i}<0} \left(\min\left(r_{k,i}, 0\right)\right)^2. \end{aligned}$$

We are now interested in the pair of retraction weights $w_{\text{retr}} = (w_{\text{APCSF}}, w_{\text{AAF}}) \in \mathbb{R}^2$ which minimizes one of the above sums

$$\min_{w_{\text{retr}}\in\mathbb{R}^2} S \quad \text{or} \quad \min_{w_{\text{retr}}\in\mathbb{R}^2} S^+.$$

While the first case enforces ideally small corrections $r_{\text{k,i}}$, the second case favors a positive protrusion component f_{prot}.

Subsequently, after estimating w_{APCSF} and w_{AAF}, we can determine w_{prot} such that the underlying protrusion process

$$X_{\text{prot}}(t_k, \theta_i) = L(t_k)\, \frac{f_{\text{prot}}(t_k, \theta_i)}{w_{\text{prot}}}$$

fulfills $\text{Var}(X_{\text{prot}}) = 1$ with $\text{Var}(\cdot)$ denoting the sample variance. This can be achieved by choosing

$$w_{\text{prot}} = \sqrt{\text{Var}(\tilde{X}_{\text{prot}})},$$

with

$$\tilde{X}_{\text{prot}}(t_k, \theta_i) := L(t_k)\, f_{\text{prot}}(t_k, \theta_i).$$

B.3 Numerical implementation of contour dynamics models

Here, we describe how a contour at time t_k can be evolved for a short time period of $[t_k, t_{k+1}]$. For an initial contour Γ_k, we start with a (preferably) equidistant set of $M \in \mathbb{N}$ contour grid points.

Further, we introduce the following vectorized notation for the spatial coordinates of these markers:

$$
\begin{aligned}
\mathbf{\Phi}_k^{(x)} &: [t_k, t_{k+1}] \to \mathbb{R}^M, &\quad t &\mapsto \mathbf{\Phi}_k^{(x)}(t),\\
\mathbf{\Phi}_k^{(y)} &: [t_k, t_{k+1}] \to \mathbb{R}^M, &\quad t &\mapsto \mathbf{\Phi}_k^{(y)}(t),\\
\mathbf{z}_k &: [t_k, t_{k+1}] \to \mathbb{R}^{2M}, &\quad t &\mapsto \mathbf{z}_k(t) := \begin{bmatrix} \mathbf{\Phi}_k^{(x)}(t) \\ \mathbf{\Phi}_k^{(y)}(t) \end{bmatrix},
\end{aligned}
$$

with corresponding center of mass trajectories $\Phi_{\text{CM}}^{(x)}(t), \Phi_{\text{CM}}^{(y)}(t) \in \mathbb{R}$. Moreover, we introduce the following vectorized notations for

- the protrusion process $\mathbf{X}_k^{\text{prot}} \in \mathbb{R}^M$ being constant for the entire time period $[t_k, t_{k+1}]$,
- the contour curvature $\boldsymbol{\kappa}(t) \in \mathbb{R}^M$,
- and the unit normal vector components $\vec{\mathbf{n}}^{(x)}(t), \vec{\mathbf{n}}^{(y)}(t) \in \mathbb{R}^M$.

Then, the vectorized model components $\mathbf{f}, \mathbf{f}_{\text{prot}}, \mathbf{f}_{\text{APCSF}}, \mathbf{f}_{\text{AAF}} : \mathbb{R}^+ \times \mathbb{R}^{2M} \to \mathbb{R}^M$ are given by

$$
\begin{aligned}
\mathbf{f}_{\text{prot}}(t, \mathbf{z}_k) &= \frac{w_{\text{prot}}}{L(t)} \mathbf{X}_k^{\text{prot}},\\
\mathbf{f}_{\text{APCSF}}(t, \mathbf{z}_k) &= -\, w_{\text{APCSF}} \left(\boldsymbol{\kappa}(t) - \frac{2\pi}{L(t)} \mathbb{1}_M \right),\\
\mathbf{f}_{\text{AAF}}(t, \mathbf{z}_k) &= -\, w_{\text{AAF}} \frac{A(t) - A_{\text{ref}}}{A_{\text{ref}} \cdot L(t)} \Big(\left(\mathbf{\Phi}^{(x)}(t) - \Phi_{\text{CM}}^{(x)}(t) \mathbb{1}_M \right) \circ \vec{\mathbf{n}}^{(x)}(t)\\
&\qquad + \left(\mathbf{\Phi}^{(y)}(t) - \Phi_{\text{CM}}^{(y)}(t) \mathbb{1}_M \right) \circ \vec{\mathbf{n}}^{(y)}(t) \Big).
\end{aligned}
$$

In this context, $\circ$ denotes the element-wise (so-called Hadamard) product and $\mathbb{1}_M$ the all-ones vector of dimension M. Now, we can rewrite Eqs. (5.14) and (5.15) for a set of M contour grid points

$$
\begin{aligned}
\frac{\partial \mathbf{z}_k(t)}{\partial t} &= \begin{bmatrix} \mathbf{f}(t, \mathbf{z}_k(t)) \circ \vec{\mathbf{n}}^{(x)}(t) \\ \mathbf{f}(t, \mathbf{z}_k(t)) \circ \vec{\mathbf{n}}^{(y)}(t) \end{bmatrix} \\
\mathbf{f}(t, \mathbf{z}_k(t)) &= \mathbf{f}_{\text{prot}}(t, \mathbf{z}_k(t)) + \mathbf{f}_{\text{APCSF}}(t, \mathbf{z}_k(t)) + \mathbf{f}_{\text{AAF}}(t, \mathbf{z}_k(t)).
\end{aligned}
\tag{B.1}
$$

Given an initial contour Γ_k with equidistant contour grid points with corresponding coordinates $\mathbf{z}_k(t_k) = \left[\mathbf{\Phi}_k^{(x)}(t_k), \mathbf{\Phi}_k^{(y)}(t_k) \right] \in \mathbb{R}^{2M}$, we compute the contour dynamics for the time interval $[t_k, t_{k+1}]$ by solving the initial value problem in Eq (B.1). For this purpose, we use the built-in `LSODA` solver in the Python package `SciPy`. This solver is based on backward differentiation formulas (BDF) – an implicit method developed to solve especially stiff ordinary differential equations, see [179, 180] for more details.

The BDF method of order 1 is better known as implicit Euler method – in our notation given by the formula:

$$\mathbf{z}_k(\tau_{i+1}) = \mathbf{z}_k(\tau_i) + h \cdot \mathbf{f}_k(\tau_{i+1}, \mathbf{z}_k(\tau_{i+1})),$$

with initial time $\tau_0 = t_k$, evaluation times $\tau_i = \tau_0 + ih$ and $\tau_N = t_{k+1}$, step size $h = \frac{\delta t}{N}$, and number of steps $N \in \mathbb{N}$. Whereas, $\mathbf{X}_k^{\text{prot}}$ is set to be constant for the above time integration, the geometric quantities $L(\tau)$, $A(\tau)$, $\boldsymbol{\kappa}(\tau)$, $\vec{\mathbf{n}}(\tau)$, and $\Phi_{\text{CM}}(\tau)$ can be updated for each iteration step $t_k \leq \tau \leq t_{k+1}$ until the coordinates $\mathbf{z}_k(\tau_N) = \mathbf{z}_k(t_{k+1})$ of the consecutive contour Γ_{k+1} are derived.

Finally, we conclude the contour evolution with a contour mapping step, i.e., we compute trajectories of all virtual markers between Γ_k and Γ_{k+1} based on a regularized flow, see Figs B.1 and B.3 for more details. Afterwards, we proceed with the contour evolution for the next time period $[t_{k+1}, t_{k+2}]$, starting again with a preferably equidistant set of contour grid points.

Regularized flows to counteract VM thinning and clustering. Since the model components from Eq (B.1) are mostly acting in normal direction, thinning and clustering effects of virtual markers are inevitable for longer time periods at the front and rear, respectively. For this reason, we propose to propagate an equidistant set of contour grid points with respect to Eq (B.1) for a short time interval $[t_k, t_{k+1}]$ only. Afterwards, we map the initial contour at time t_k to the next one at time $t_{k+1} = t_k + \delta t$ based on a regularized flow as described in [DS1]. The extent of the regularization of ϕ_k is controlled by a single parameter $\lambda_{\text{reg}} \geq 0$.

In Fig B.3, we illustrate the two different kinds of marker trajectories: (1) the contour propagation based (green dashed lines) and (2) the contour mapping (blue dashed lines) under which X_{prot} is transported. By comparing both trajectories, we nicely see that a regularization, under which X_{prot} is transported, is required to avoid thinning/clustering effects. Moreover, we use this contour mapping for the underlying spatio-temporal coordinate system of the kymograph descriptions displayed later on. More precisely, horizontal lines of any later shown kymograph describe the corresponding quantity along regularized VM trajectories.

In B.1, we summarized the above algorithm in form of a pseudocode.

Algorithm 1: ContourDynamicsModel

```
Input: $\Gamma_0,\ \delta t > 0,\ \lambda_{\text{reg}} \gg 0,\ \vartheta_{\text{model}} \in \mathbb{R}^4,\ \vartheta_{\text{prot}} \in \mathbb{R}^{n_{\text{prot}}}$
Output: $\Gamma_0, \dots, \Gamma_{K-1}$

/* Generate protrusion process for the entire time span */
$X_0^{\text{prot}}, \dots, X_{K-2}^{\text{prot}}$ = StochasticProcess($\vartheta_{\text{prot}}$);

for $k = 0$ to $K - 2$ do
    /* Compute virtual marker distance rate */
    VMDR = VMDistanceRate($\Gamma_k$);

    /* Adjust protrusion process w.r.t VMDR */
    $X_k^{\text{prot}} = X_k^{\text{prot}}/\text{VMDR}$;

    /* Compute coordinates of equidistant contour grid points */
    $p_{\text{init}}$ = GaussianProcessRegression($\Gamma_k$);

    /* Propagate contour grid points based on model functions */
    $p_{\text{end}}$ = SolveIVP($p_{\text{init}}, \delta t, X_k^{\text{prot}}, \vartheta_{\text{model}}$);

    /* Compute contour mapping based on regularized flow */
    $\Gamma_{k+1}$ = RegularizedFlow($\Gamma_k,\ p_{\text{end}}, \lambda_{\text{reg}}$);
end
return $\Gamma_0, \dots, \Gamma_{K-1}$
```

Figure B.1. **Algorithm to perform contour dynamics model.** The algorithm input consists of an initial contour Γ_0, a step width $\delta t > 0$, a regularization parameter λ_{reg}, the model parameters $\vartheta_{\text{model}} = \{w_{\text{prot}}, w_{\text{APCSF}}, w_{\text{AAF}}, A_{\text{ref}}\}$, and a set of n_{prot} parameters regarding the stochastic protrusion process denoted with $\vartheta_{\text{prot}} \in \mathbb{R}^{n_{\text{prot}}}$. The output contains the artificial cell track based on consecutive contours $\Gamma_0, \dots, \Gamma_K - 1$.

B.4 Model equations with relative weights

In Eq (5.16), our model is notated with absolute weights $w_{\text{prot}}, w_{\text{APCSF}}, w_{\text{AAF}} > 0$. Alternatively, with an overall velocity parameter

$$w_f = w_{\text{prot}} + w_{\text{APCSF}} + w_{\text{AAF}} \cdot 1\mu m$$

our model can be notated with resoect to relative weights:

$$\begin{aligned} r_{\text{prot}} &= \frac{w_{\text{prot}}}{w_f}, \\ r_{\text{APCSF}} &= \frac{w_{\text{APCSF}}}{w_f}, \\ r_{\text{AAF}} &= \frac{w_{\text{AAF}} \cdot 1\mu m}{w_f} = (1 - r_{\text{prot}} - r_{\text{APCSF}}). \end{aligned}$$

Then, we can rewrite Eqs (5.15) and (5.16) as

$$f = w_f \cdot (f_{\text{prot}} + f_{\text{APCSF}} + f_{\text{AAF}}),$$

with the following three components:

$$
\begin{aligned}
I: &\quad f_{\text{prot}}(t,\theta) = r_{\text{prot}} \frac{X_{\text{prot}}(t,\theta)}{L(t)}, \\
II: &\quad f_{\text{APCSF}}(t,\theta) = r_{\text{APCSF}} \left(\frac{2\pi}{L(t)} - \kappa(t,\theta) \right), \\
III: &\quad f_{\text{AAF}}(t,\theta) = (1 - r_{\text{prot}} - r_{\text{APCSF}}) \frac{A(t) - A_{\text{ref}}}{A_{\text{ref}} \cdot L(t)} \langle \Phi(t,\theta) - \Phi_{\text{CM}}(t), \vec{n}(t,\theta) \rangle .
\end{aligned}
$$

B.5 Ornstein-Uhlenbeck process as protrusion process

In this section, we define the protrusion process in our model as diffusion process, more precisely as an Ornstein-Uhlenbeck process which is denoted by

$$X : [0,T] \times [0,2\pi) \to \mathbb{R}, \quad (t,\theta) \mapsto X(t,\theta).$$

Further, we introduce the notation for discrete time steps

$$X(k\delta t, \theta) = X(t_k, \theta) = X_k(\theta),$$

with $X(0,\theta) = X_0(\theta) = 0$ for all $\theta \in [0,2\pi)$.

In the following, we introduce several functions underlying our Ornstein-Uhlenbeck process. Afterwards, we present the exact definition of X.

Scaling functions. An Ornstein-Uhlenbeck process consists of a mean reversion term, preventing the process from getting too small or too large, and a diffusion term, resulting in process changes due to stochastic innovations. For both terms, we introduce the scaling functions respectively:

- mean reversion rate function:

$$a : [0,T] \times [0,2\pi) \to \mathbb{R}^+ \text{ with } a_k(\theta) := a(t_k,\theta), \tag{B.2}$$

- diffusion rate function:

$$b : [0,T] \times [0,2\pi) \to \mathbb{R}^+ \text{ with } b_k(\theta) := b(t_k,\theta). \tag{B.3}$$

Innovation function. As underlying correlation function, from which we draw innovations, we use the normalized Poisson kernel:

$$\tilde{k}_r(\theta,\theta') = \frac{k_r(\theta,\theta')}{\sqrt{k_r(\theta,\theta)k_r(\theta',\theta')}} = \frac{(1-r)^2}{1 - 2r\ \cos(\theta - \theta') + r^2} \tag{B.4}$$

with $\theta, \theta' \in [0,2\pi),\ r \in [0,1)$. It holds the following properties: $\left(\frac{1-r}{1+r}\right)^2 \leq \tilde{k}_r(\theta,\theta') < 1$ for all $\theta \neq \theta'$ and $\tilde{k}_r(\theta,\theta') = 1$ if and only if $\theta = \theta'$. In Fig B.4, the Poisson kernel function from Eq (5.1) and its normalized version from Eq (B.4) are shown for different parameters $r \in [0, 1)$.

Then, the innovations of the Ornstein-Uhlenbeck process are denoted by η and are realized by the following Gaussian process:

$$\begin{aligned} \eta : [0, T] \times [0, 2\pi) &\to \mathbb{R}, \\ (t, \theta) &\mapsto \eta_t(\theta), \\ (\eta)_t &\sim \mathcal{GP}(0, \tilde{k}_{r_{\text{inn}}}(\cdot, \cdot)), \end{aligned}$$

with zero mean and covariance function $\tilde{k}_{r_{\text{inn}}}(\cdot, \cdot)$ from Eq (B.4) and innovation bandwidth parameter $r_{\text{inn}} \in (0, 1)$. For the discrete-time case, we define similarly

$$\eta_k(\theta) := \eta(t_k, \theta). \tag{B.5}$$

By choosing a normalized Poisson kernel as covariance function, we obtain spatially correlated noise, while not affecting the overall variance of X_k. In this context, we use the property that $\tilde{k}_{\text{inn}}(\theta, \theta) = 1$ for all $\theta \in [0, 2\pi)$.

Discrete formulation. Now, we present a time-discrete formula in order to generate an Ornstein-Uhlenbeck process containing the functions from Eqs (B.2), (B.3), and (B.5). Again, we choose a Lagrangian reference frame to describe the process X_k. For a virtual marker labeled by an initial (normalized) arc length coordinate $\theta \in [0, 2\pi)$, the evolution of the protrusion process along this virtual marker is given by $X_k(\theta)$ and starts at $X_0(\theta) = 0$. More precisely, X_k is defined iteratively by the following Ornstein-Uhlenbeck process:

$$\begin{aligned} X_{k+1}(\theta) &= X_k(\theta) + \delta t \cdot \Delta X_k(\theta), \\ \Delta X_k(\theta) &= -a_k(\theta) \cdot X_k(\theta) + \sqrt{2 a_k(\theta)} \cdot b_k(\theta) \cdot \eta_k(\theta). \end{aligned} \tag{B.6}$$

In the following, we will use a simplified version with constant rates $a_k(\theta) = a \in \mathbb{R}^+$ and $b_k(\theta) = b \in \mathbb{R}^+$:

$$\Delta X_k(\theta) = -a \cdot X_k(\theta) + \sqrt{2a} \cdot b \cdot \eta_k(\theta). \tag{B.7}$$

Since the magnitude of X_k can be also adjusted by the protrusion weight w_{prot} later on, we recommend to set $b = 1$. This way, and by choosing the normalized Poisson kernel as covariance function of η_k, we obtain $X_k \sim \mathcal{N}(0, 1)$.

Polarization function. A cell polarization can be easily added to the model. This polarization can be induced, e.g., by an extracellular nutrient gradient or by a "leading process" defining the general tendency of the cell to move persistently in one direction. We assume that the polarization of the cell is time-dependent and defined by a singular location on the cell contour representing the mid of the cell's front. From these assumptions, we introduce the following function

$$\theta_{\text{pol}} : [0, T] \to [0, 2\pi), \quad t \mapsto \theta_{\text{pol}}(t).$$

A simple test case for a constant polarization would be to choose $\theta_{\text{pol}}(\cdot) = \pi$ as we did for Hawkes processes in Chapter 5.

Further, we define the polarization function $p(t, \theta)$ as the following mapping:

$$\begin{aligned} p : [0, T] \times [0, 2\pi) &\to \mathbb{R}^+ \text{ with} \\ (t, \theta) &\mapsto \tilde{k}_{r_{\text{pol}}}(\theta, \theta_{\text{pol}}(t)), \end{aligned}$$

with normalized Poisson kernel $\tilde{k}_{r_{\text{pol}}}$ defined as in Eq (B.4) with corresponding polarization parameter $r_{\text{pol}} \in [0, 1)$. The polarization function fulfills the property $\left(\frac{1-r_{\text{pol}}}{1+r_{\text{pol}}}\right)^2 \leq p_k(\theta) \leq 1$ and reaches its maximum $p_k(\theta) = 1$ if and only if $\theta = \theta_{\text{pol}}(t_k)$, for more details see Fig B.4.

For the case $r_{\text{pol}} = 0$ and any function $\theta_{\text{pol}} : [0, T] \to [0, 2\pi)$, the polarization function simplifies to

$$p(\cdot, \cdot) = \tilde{k}_0(\cdot, \theta_{\text{pol}}(\cdot)) = 1,$$

i.e., no polarization effect takes place. For the case $r_{\text{pol}} \to 1$, the polarization functions simplifies to

$$p(t, \theta) = \begin{cases} 1, & \theta = \theta_{\text{pol}}(t) \\ 0, & \theta \neq \theta_{\text{pol}}(t) \end{cases}$$

which is equal to the indicator function $\mathbb{1}_{\{\theta_{\text{pol}}(t)\}}(\theta)$. Finally, we introduce a discrete notation:

$$p_k(\theta) := p(t_k, \theta) = \tilde{k}_{r_{\text{pol}}}(\theta, \theta_{\text{pol}}(t_k)). \tag{B.8}$$

Protrusion component. Motivated from the biological insight that protrusions and retractrions are triggered by the underlying actin and myosin concentration, respectively, we designed our model to separate the formation of protrusions from the formation of retractions. However, in order to obtain a separation of protrusions, resulting from the protrusion component, and retractions, resulting from the APCSF and AAF, the process $X(t, \theta)$ should only take positive values. For this reason, we propose the following modifications of the Ornstein-Uhlenbeck process from Eq (B.6):

$$\begin{aligned} X^{+}_{\text{prot}}(t_k, \theta) &:= \frac{p_k(\theta)}{\text{VMDR}(t_k, \theta)} \cdot X_k(\theta)^+, \qquad \text{(B.9)} \\ X^{2}_{\text{prot}}(t_k, \theta) &:= \frac{p_k(\theta)}{\text{VMDR}(t_k, \theta)} \cdot X_k(\theta)^2, \\ X^{\exp}_{\text{prot}}(t_k, \theta) &:= \frac{p_k(\theta)}{\text{VMDR}(t_k, \theta)} \cdot \exp\left(X_k(\theta)\right), \\ X^{\text{lin}}_{\text{prot}}(t_k, \theta) &:= \frac{p_k(\theta)}{\text{VMDR}(t_k, \theta)} \cdot X_k(\theta) + c_{\text{lin}} \\ X^{\log}_{\text{prot}}(t_k, \theta) &:= \frac{p_k(\theta)}{\text{VMDR}(t_k, \theta)} \cdot \frac{1}{1 + \exp\left(-\beta_{\log} X_k(\theta)\right)}, \end{aligned}$$

with $c_{\text{lin}} > 0$, $\beta_{\text{log}} > 0$, a polarization function p_k defined as in Eq (B.8), and virtual marker distance ratio defined as in Eq (5.4). For the linear shift modification, $c_{\text{lin}} > 0$ needs to be sufficiently large in order to obtain mostly positive values for $X_{\text{prot}}^{\text{lin}}$.

Simulating amoeboid motility by an Ornstein-Uhlenbeck process

In Chapter 5, we have shown that amoeboid cell motility can be well simulated with a self-exciting Poisson point process, as so-called Hawkes process. In this section, we simulate contour dynamics driven by an Ornstein-Uhlenbeck process (OUP) defined as in Eq (B.7). The OUP is commonly applied in other cell motility models where it is often used to model changes of two-dimensional bio marker concentrations. For this reason, we investigated if the OUP is also suitable within our model. In the above section, we introduced multiple modifications of the OUP, necessary to separate the formation of protrusions from the formation of retractions. We therefore studied the influence of each modification on the overall contour dynamics.

First, we generated cell tracks based on the OUP in Eq (B.7) for all modifications mentioned above. The mean reversion rate and diffusion rate were set to be constant: $a = 0.05$ and $b = 1$. Furthermore, we assumed no polarization, i.e., $r_{\text{pol}} = 0$. Since the modifications from Eq (B.9) affect the magnitude of the protrusion process in different ways, the protrusion weight w_{prot} needed to be adjusted for each modification. For this reason, we have chosen $w_{\text{prot}} = 15; 5; 5; 7.5; 20\mu m/s$ for X_{prot}^{+}, X_{prot}^{2}, $X_{\text{prot}}^{\text{exp}}$, $X_{\text{prot}}^{\text{lin}}$, and $X_{\text{prot}}^{\text{log}}$, respectively. In general, the reference area of A_{ref} acts as lower bound of the area in our model. From experimental recordings, we can say that $A_{\text{ref}} = 80\mu m^2$ is plausible for an exemplary cell track. The exact choice of all parameters can be found in Table B.1. The first group of parameters regarding the contour parametrization and the flow between contours is described in a more detailed way in [DS1].

In the top row of Fig B.2, cell tracks based on the different modifications from Eq (B.9) are shown. For each modification, the same realization of the Ornstein-Uhlenbeck process X_{prot} is underlying. The non-polarized behavior of the cell tracks can be clearly seen. From the covered trace (grey area) of each cell track and its corresponding center of mass trajectory (colored line), we observe different degrees of motility for $X_{\text{prot}}^{\text{exp}}$, X_{prot}^{+}, X_{prot}^{2}, $X_{\text{prot}}^{\text{lin}}$, and $X_{\text{prot}}^{\text{log}}$ (descending order).

Furthermore, the kymographs of the overall local motion f and its three components $f_{\text{prot}}, f_{\text{APCSF}}, f_{\text{AAF}}$ are displayed for each of these cell tracks. For X_{prot}^{+}, the expanding regions (red, first two columns) are more distinct and with sharp edges directly resulting from the non-smooth behavior of X_{prot}^{+}. In contrast, the expanding regions of X_{prot}^{2} are smoother. Furthermore, by taking also the negative parts of X_{prot} into account, the number of expanding regions approximately doubles. For the third case $X_{\text{prot}}^{\text{exp}}$, we observe kymographs similar to X_{prot}^{+} but with smoother transitions between expansions and retractions. In the fourth case, we have chosen $c_{\text{lin}} = 1.96$ such that $\mathbb{P}(X_{\text{prot}}^{\text{lin}} > 0) = 0.975$. As a consequence of this mean shift, the corresponding

Table B.1. **Choice of parameters and meaning.**

Parameter	Value	Unit	Meaning
Contour parametrization			
r_{cont}	0.6	–	GPR smoothing
σ_{noise}	0.05	–	GPR noise
λ_{reg}	10	$\frac{\mu m^2}{s^2}$	Flow regularization
A_{ref}	80	μm^2	Reference area
Ornstein-Uhlenbeck process			
a	0.05	s^{-1}	Mean reversion rate
b	1	$s^{-\frac{1}{2}}$	Diffusion rate
r_{inn}	0.5	–	Corr. length of innovations
r_{pol}	0	–	Corr. length of polarization
Model weights			
w_{prot}	5–20	$\frac{\mu m}{s}$	Protrusion weight
w_{APCSF}	0.1	$\frac{\mu m}{s}$	APCSF weight
w_{AAF}	1	$\frac{\mu m}{s}$	AAF weight

List of parameters for simulated contour dynamics based on an Ornstein-Uhlenbeck process.

protrusion kymograph shows larger expanding regions. Due to a larger protrusion component, the size of the cell contour is significantly increased. Since f_{APCSF} is proportional to the contour curvature, which decreases due to a larger contour size, the corresponding kymograph (3rd column) shows less pronounced curvature patterns. An increased contour area can be also inferred from the f_{AAF} kymograph (4th column), indicated by a dark blue color, leading to a stronger area adjustment. For the logistic modification $X_{\text{prot}}^{\text{log}}$ with $\beta_{\text{log}} = 2$, a similar effect of an increased contour size can be observed.

In Vid B.3, all five cell tracks are displayed at tenfold speed. In accordance to the kymographs from Fig B.2, we observed the non-smooth behavior for X_{prot}^{+} with abrupt changes of the cell contour. For X_{prot}^{2}, we observed more protrusions evolving more smoothly compared to X_{prot}^{+}. However, by creating too many protrusions, canceling each other out, the overall cell motility can be reduced. This effect was also shown in [15]. Since X_{prot}^{2} depends also on the negative regions of the initial Ornstein-Uhlenbeck process, the resulting cell track differs significantly from the other four cell tracks, which are synchronous most of the time. From the exponential modification $X_{\text{prot}}^{\text{exp}}$, a cell track with rapidly generating protrusions was observed. Finally, the cell tracks obtained from $X_{\text{prot}}^{\text{lin}}$ and $X_{\text{prot}}^{\text{log}}$ were almost stationary with an increased cell shape and lesser curvature which coincides with the above observations.

In general, we recommend using the exponential modification $X_{\text{prot}}^{\text{exp}}$ resulting in smooth contour changes driven by fast and more explorative protrusions. However,

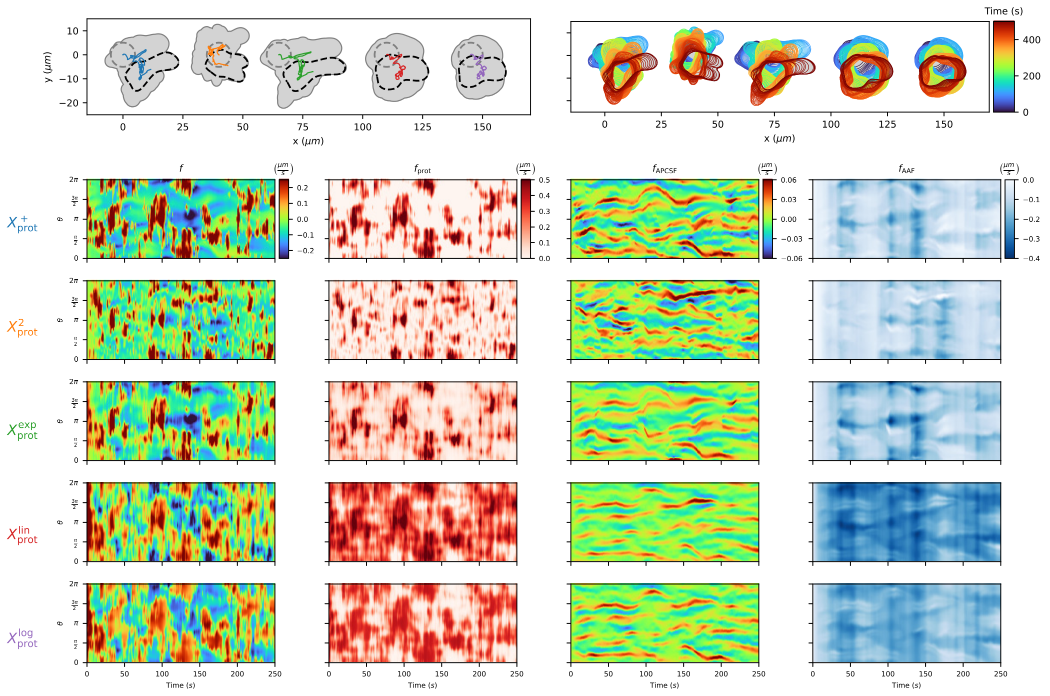


Figure B.2. Comparison of cell tracks obtained from different modifications of the underlying Ornstein-Uhlenbeck process. In the first row, the entire area (trace) covered by each cell track is displayed (left, gray area) as well as their center of mass trajectories (left, colored lines) and the evolution of the contour dynamics over a time span of $T = 500s$ (right, colored contours). Below, for each modification, the corresponding kymographs of the following quantities are displayed for the first half of each cell track: the local motion f and its three components f_{prot}, f_{APCSF}, and f_{AAF} (from left to right).

some protrusions look artificial as well as additional artifacts such as a pulsating membrane and a partially swimming type of locomotion. While this method provides a simple way of generating cell tracks, they can be easily distinguished from experimental cell tracks. Based on the findings of Chapter 5, where we have chosen a Hawkes process as underlying protrusion component, much more realistic contour dynamics were generated. Due to its self-exciting property, the Hawkes process is capable of producing cascades of multiple protrusions with accompanying reorientation phases of the cell. Compared to the OUP, the Hawkes process is therefore the better choice for modeling amoeboid cell motility.

B.6 Supporting figures

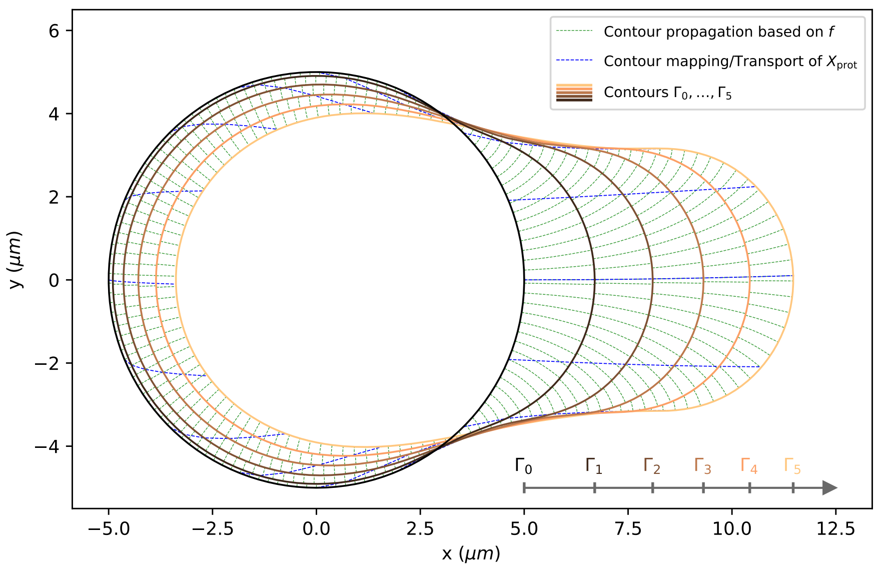


Figure B.3. Artificial contour dynamics based on our model with comparison of underlying contour propagation and contour mapping. First, the contour is propagated by the model function f evaluated for an equidistant set of grid points on the contour (green dashed trajectories). Secondly, the protrusion component X_{prot} is propagated by a strongly regulrized flow to avoid thinning and clustering effects of virtual markers (blue dashed tracetories). This contour mapping can then be used as underlying coordinate system in kymograph representations.

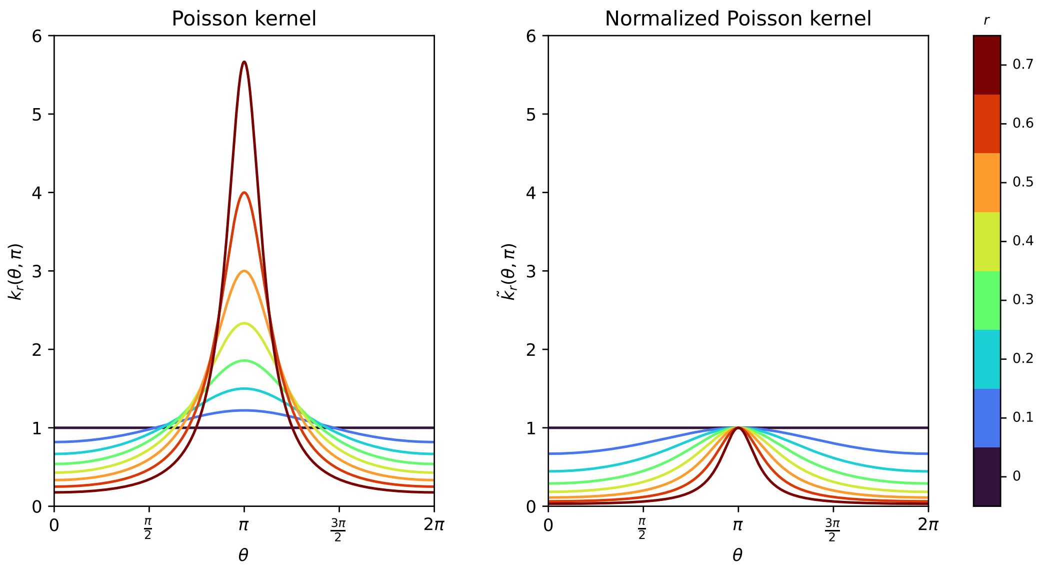


Figure B.4. Comparison of Poisson kernel function and its normalized version. Poisson kernel function k_r (left) and normalized Poisson kernel function $\tilde{k}_r$ (right)for varying radius parameter $r \in \{0, 0.1, \ldots, 0.7\}$.

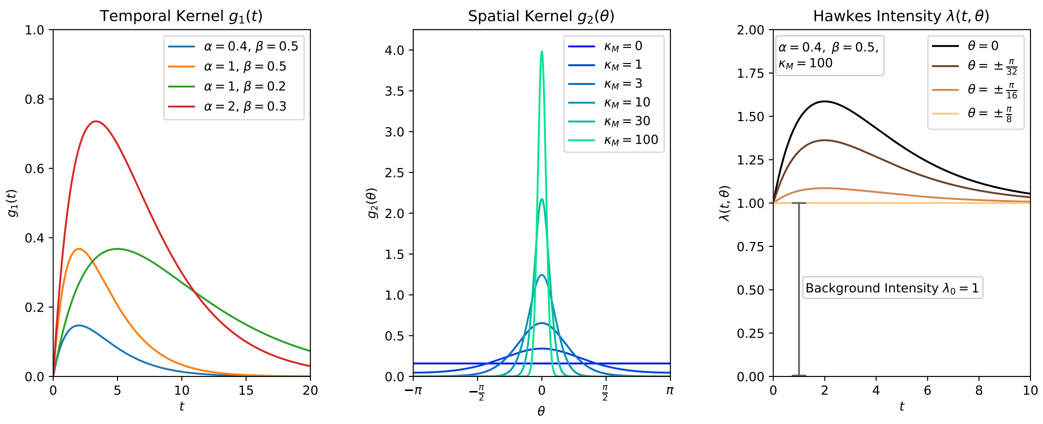


Figure B.5. Kernel functions used to generate artificial cell tracks driven by a Hawkes process. Temporal kernel $g_1(t)$ with varying arrival intensity $\alpha > 0$ and exponential decay rate $\beta > 0$ (left). Spatial Kernel $g_2(\theta)$ with varying concentration parameter $\kappa_M \geq 0$ (middle). Hawkes Intensity $\lambda(t, \theta)$ with background intensity $\lambda_0 = 1$ for varying θ along the cell contour and the following choice of parameters: $\alpha = 0.4$, $\beta = 0.5$, and $\kappa_M = 100$ as used in our model (right).

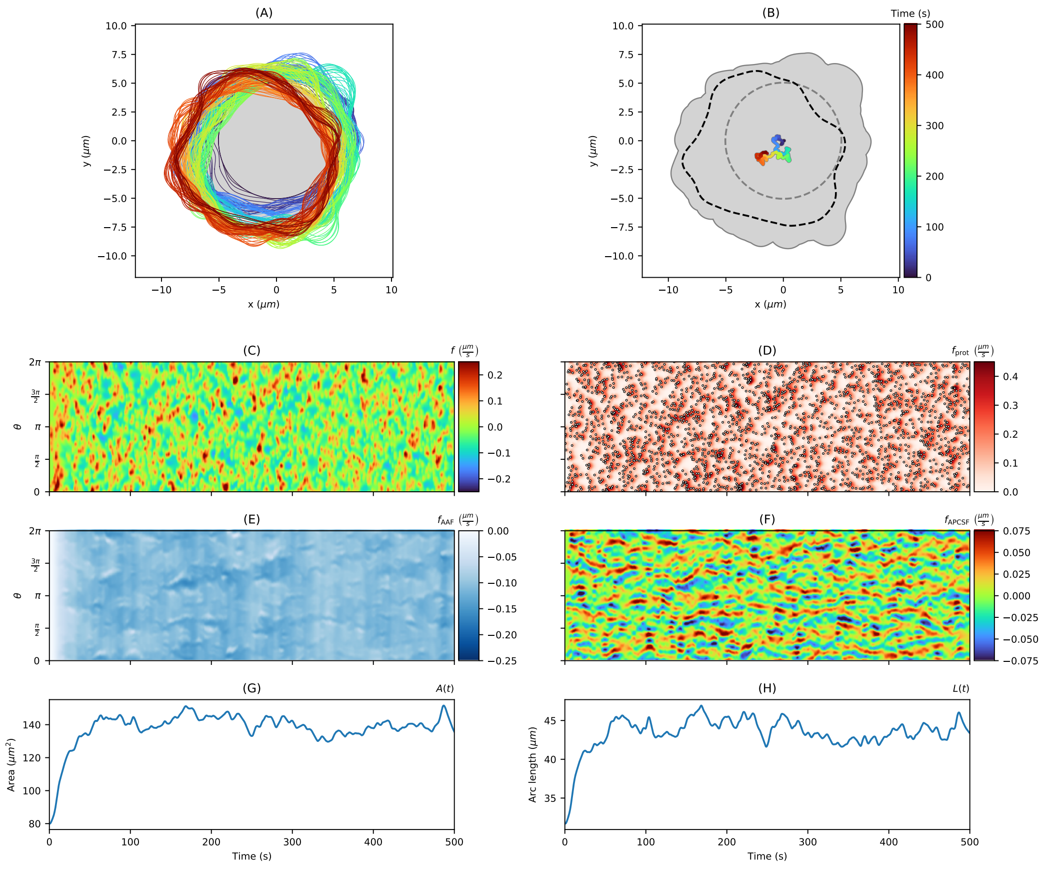


Figure B.6. Non-polarized cell track based on a Poisson point process as protrusion process and the corresponding kymographs displayed as in Fig 5.3, see File B.1 for additional contour dynamics based on a Poisson point process.

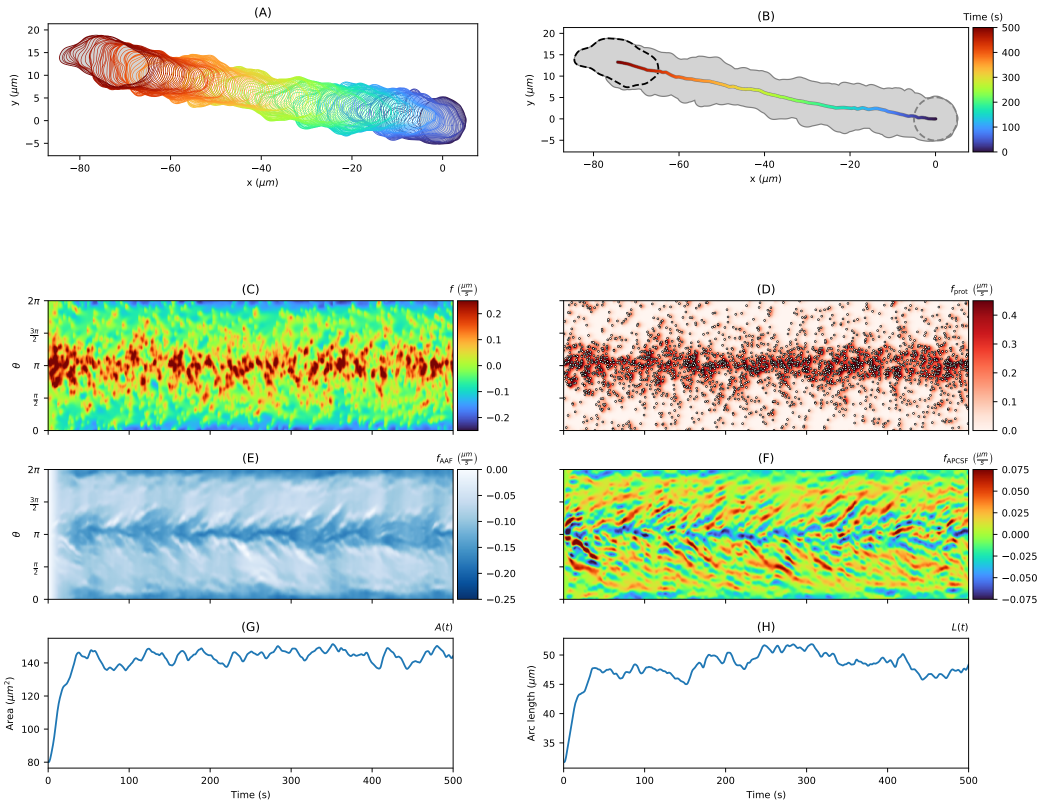


Figure B.7. Polarized cell track based on a Poisson point process as protrusion process and the corresponding kymographs displayed as in Fig 5.4, see File B.2 for additional contour dynamics based on a Poisson point process.

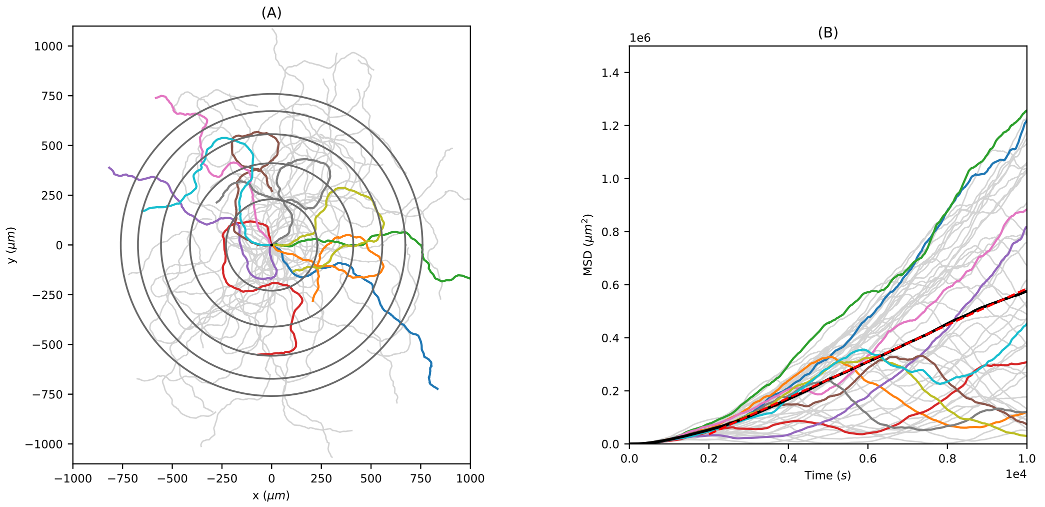


Figure B.8. Long-term diffusion analysis of polarized artificial cell tracks based on a Hawkes process. Long-term diffusion analysis of polarized artificial cell tracks based on a Hawkes process. **(A)** Center of mass trajectories of 50 polarized cell tracks over a longer time period $T = 10000s$ (colored and gray lines). The root mean squared displacement (RMSD) is shown for every $2000s$ (gray circles). **(B)** Corresponding mean squared displacement (MSD, bold black line) with linear fit (dashed red line).

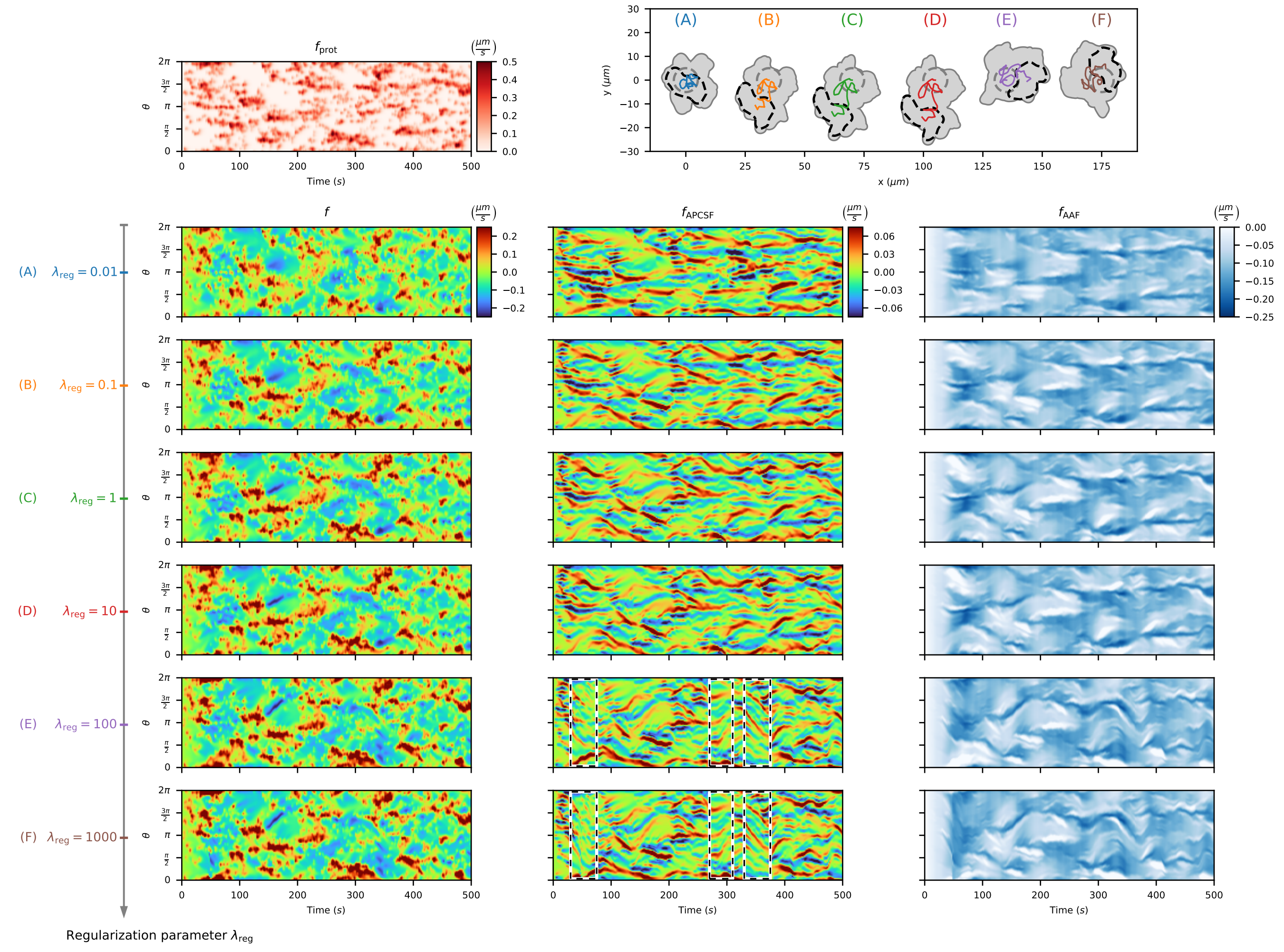


Figure B.9. Parameter study with varying regularization parameter λ_{reg} during non-polarized cell motility. For each cell track the same protrusion component is underlying (top left). The center of mass trajectory (colored lines) as well as the covered area of each cell track (gray area) are displayed (top right). Based on different regularization schemes, different kymographs are computed: Local motion (left column), the APCSF component (middle column), and the AAF component (right column).

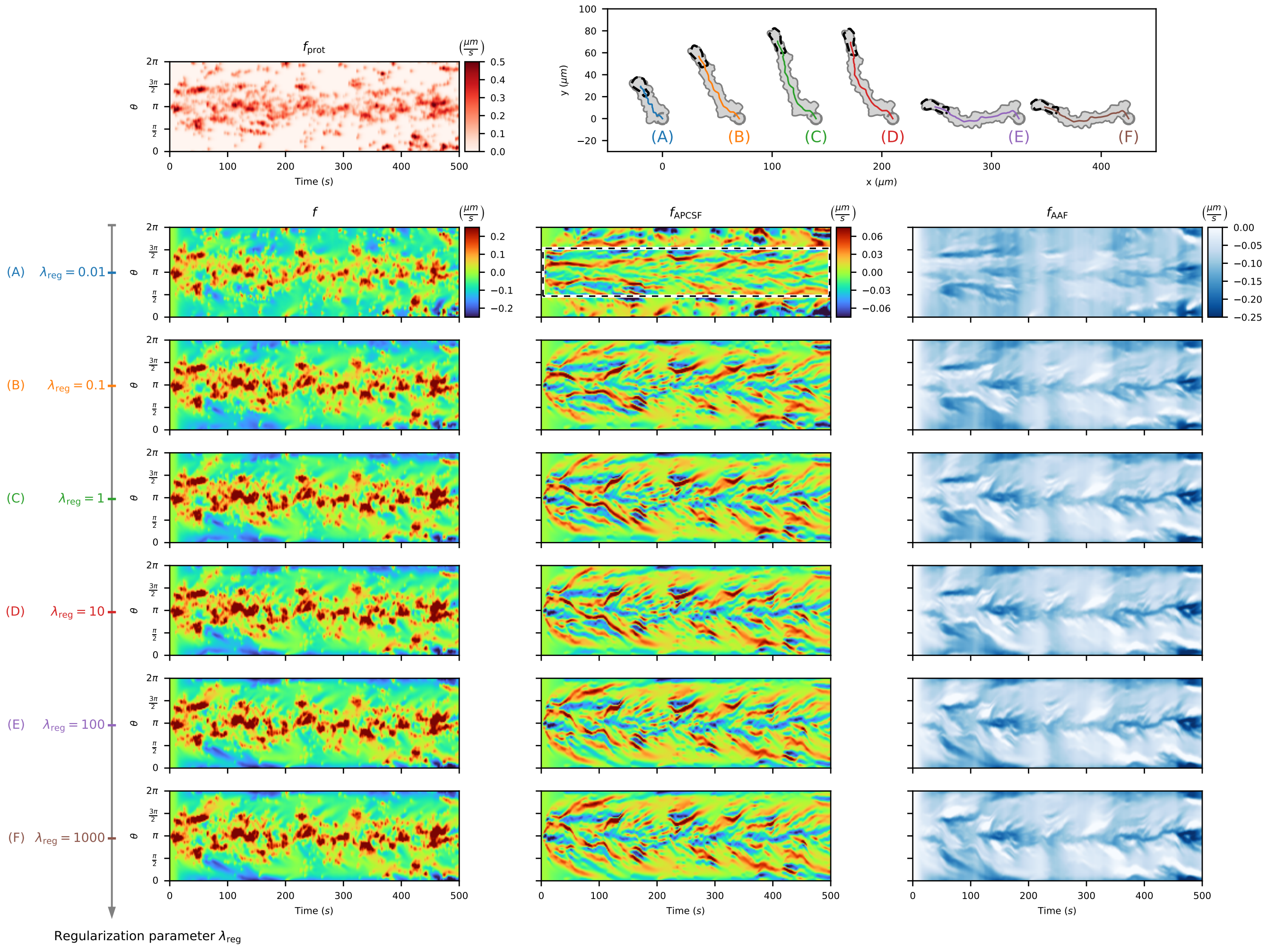


Figure B.10. Parameter study with varying regularization parameter λ_{reg} during polarized cell motility. For each cell track the same protrusion component is underlying (top left). The center of mass trajectory (colored lines) as well as the covered area of each cell track (gray area) are displayed (top right). Based on different regularization schemes, different kymographs are computed: Local motion (left column), the APCSF component (middle column), and the AAF component (right column).

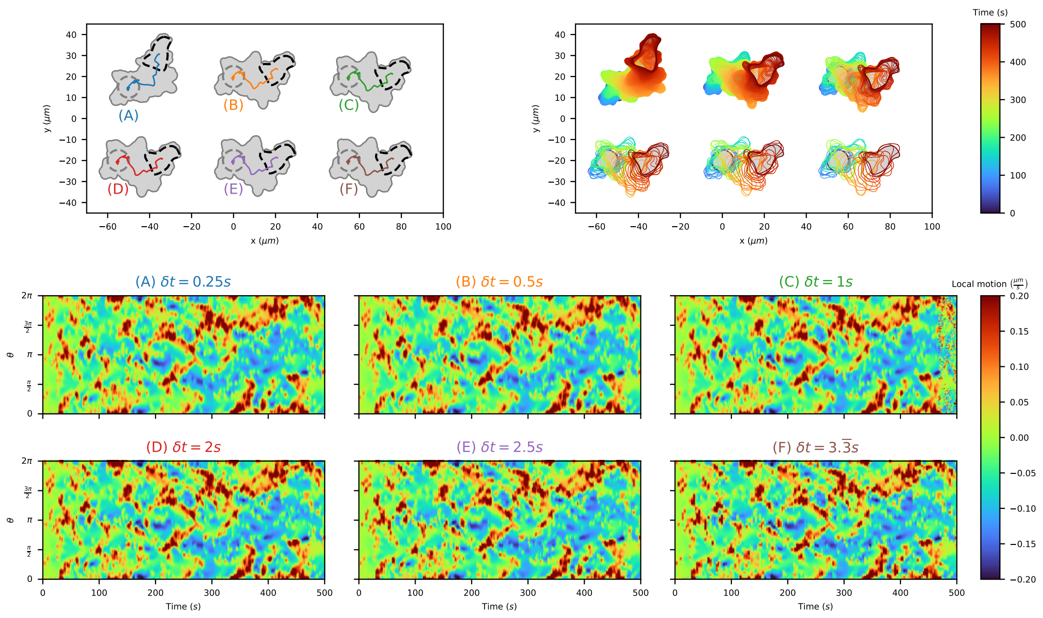


Figure B.11. Comparison of local motion kymographs of simulated non-polarized cell tracks for different temporal resolutions: $\delta t \in \{0.25, 0.5, 1, 2, 2.5, 3.\overline{3}\}$. In the top left corner, the center of mass trajectory (colored lines) and the trace of each cell track (gray area) are displayed. In the top right corner, the contour dynamics (colored lines) and the trace of the cell track (gray area) are shown. Below, kymographs of the local motion based on our model are listed for each cell track.

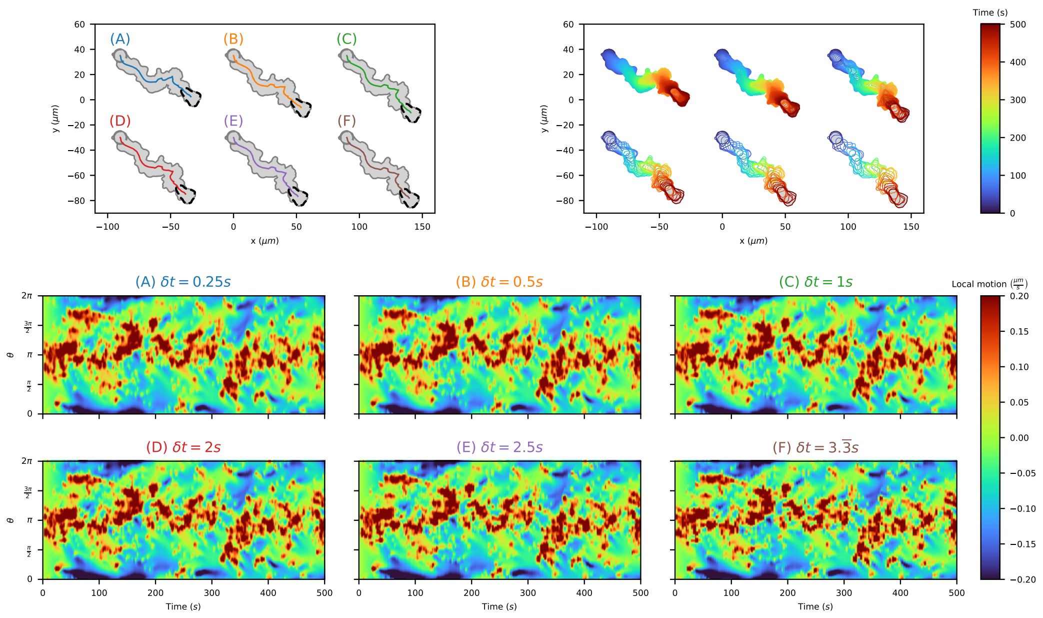


Figure B.12. Comparison of local motion kymographs of simulated polarized cell tracks for different temporal resolutions: $\delta t \in \{0.25, 0.5, 1, 2, 2.5, 3.\overline{3}\}$. In the top left corner, the center of mass trajectory (colored lines) and the trace of each cell track (gray area) are displayed. In the top right corner, the contour dynamics (colored lines) and the trace of the cell track (gray area) are shown. Below, kymographs of the local motion based on our model are listed for each cell track.

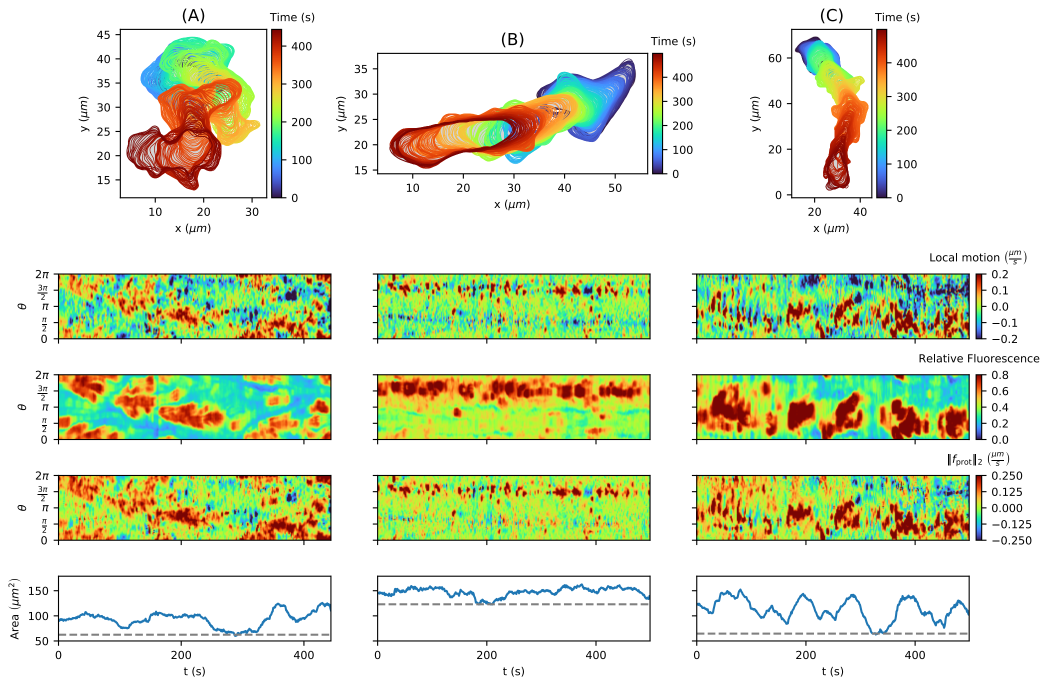


Figure B.13. Inference of protrusion component and model weights of (experimental) amoeboid cell tracks. Collection of three *D. discoideum* tracks driven by the standard amoeboid type of cell motion with corresponding kymographs: Local motion, relative fluorescence intensity, the underlying protrusion component inferred from our model, and the contour area. For each cell track, the 1st percentile of the individual area time series (last row) was chosen as the reference area (gray dashed line) in our model: 62.43, 122.99, and $64.61\mu m^2$, respectively. The model weights $(w_{\text{prot}},\ w_{\text{APCSF}},\ w_{\text{AAF}})$ were estimated individually for each cell track: $(4.513,\ 0.023,\ 0.902)$, $(4.513,\ 0.048,\ 1.429)$, and $(6.505,\ 0.070,\ 0.676)$, respectively from **(A)** to **(C)**.

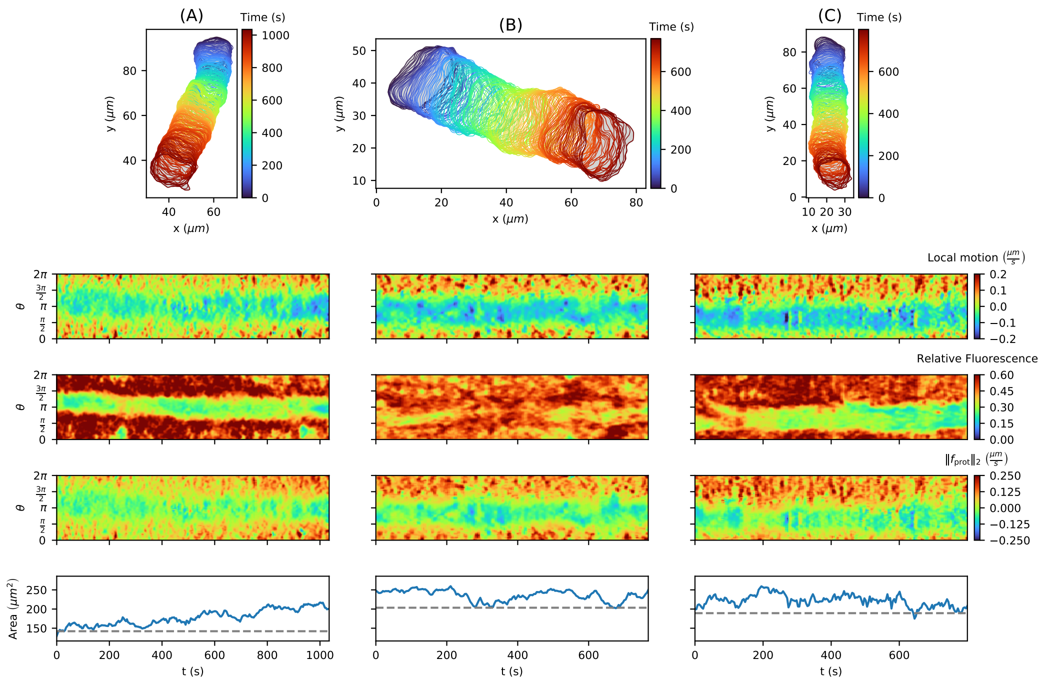


Figure B.14. Inference of protrusion component and model weights of (experimental) fan-shaped cell tracks. Collection of three *D. discoideum* tracks driven by a fan-shaped type of cell motion with corresponding kymographs: Local motion, relative fluorescence intensity, the underlying protrusion component inferred from our model, and the contour area. For each cell track, the 1st percentile of the individual area time series (last row) was chosen as the reference area (gray dashed line) in our model: 142.15, 203.45, and $189.24\mu m^2$, respectively. The model weights $(w_{\text{prot}}, w_{\text{APCSF}}, w_{\text{AAF}})$ were estimated individually for each cell track: $(3.121, 0.000, 0.911)$, $(4.485, 0.000, 1.844)$, and $(4.428, 0.000, 1.760)$, respectively from **(A)** to **(C)**.

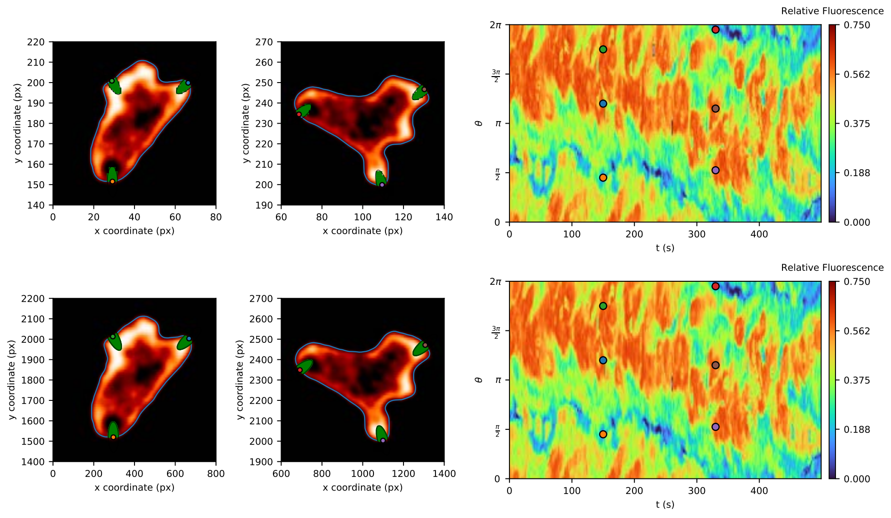


Figure B.15. **Computation of relative fluorescence intensity for experimental microscopy data** and tenfold upsampled data via ellipses along the cell contour. **(Top row)** Two exemplary frames of the original microscopy data from which the segmented contour (blue) is derived. The relative fluorescence intensity averaged over ellipses along the cell contour (green) are then translated into a kymograph (right column). The position of six exemplary virtual markers on the cell contour and in the kymograph are displayed as colored dots. **(Bottom row)** Tenfold upsampled image data with corresponding kymograph which shows almost no deviations from the original kymograph.

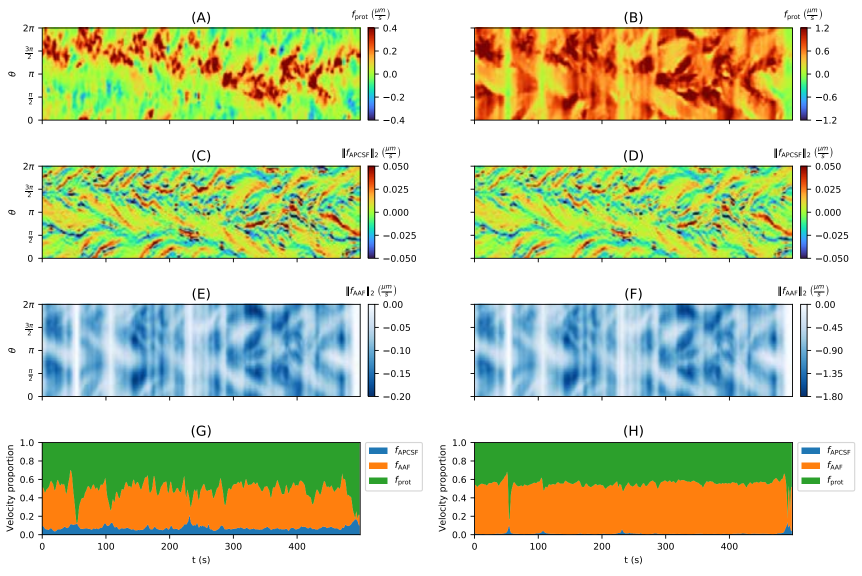
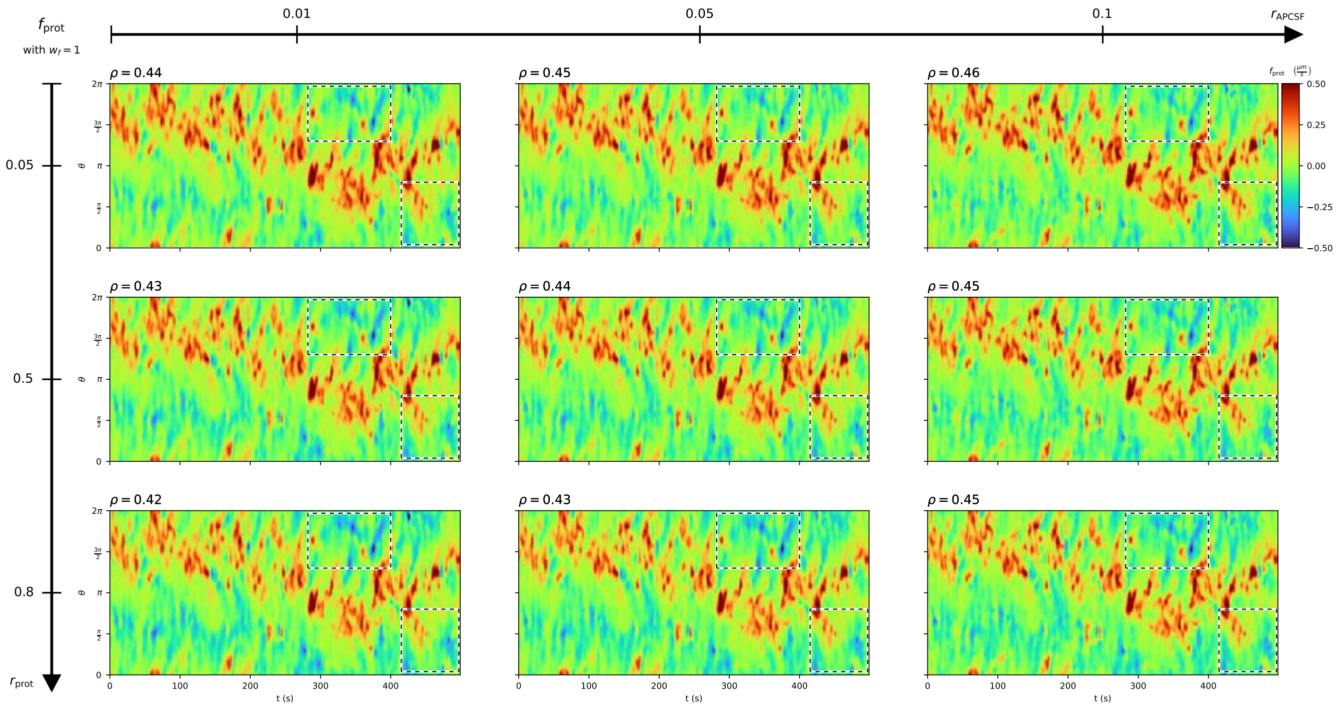
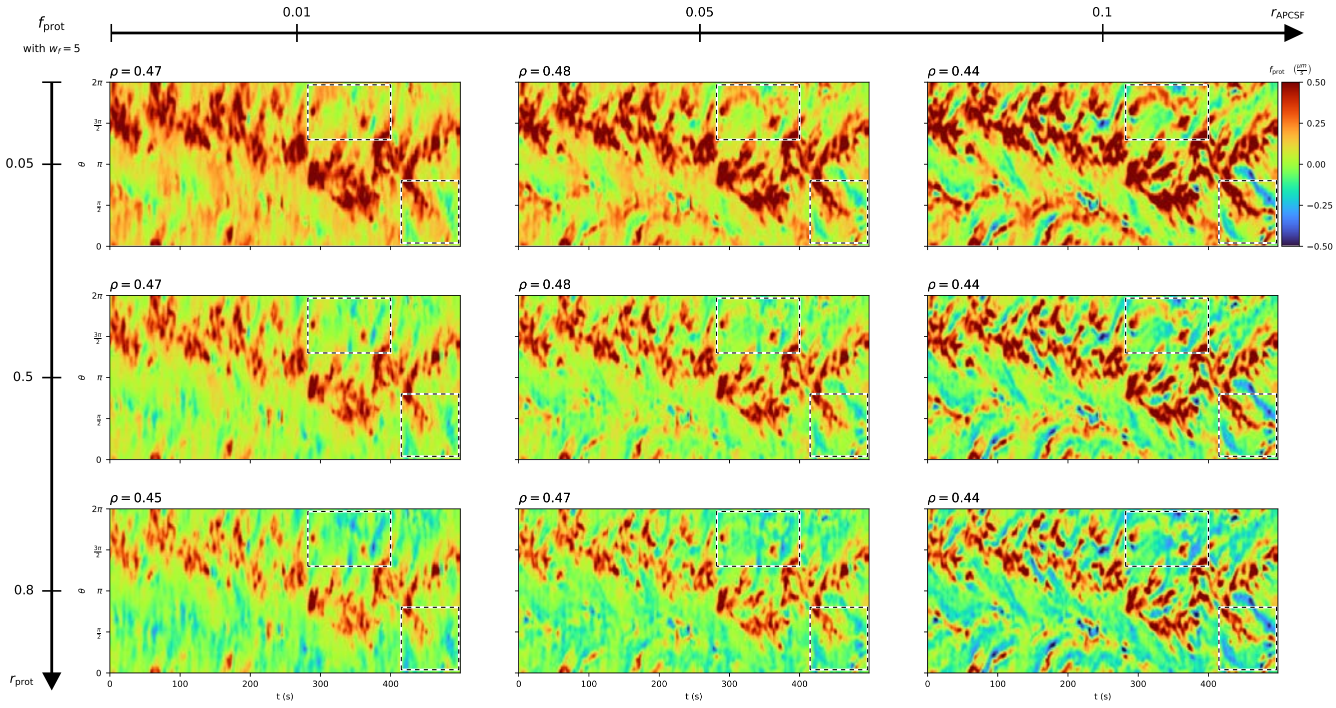


Figure B.16. **Model components extracted from experimental cell track** of Fig 5.7 for two pairs of model weights $(w_{\text{prot}}, w_{\text{APCSF}}, w_{\text{AAF}})$:
$(6.634,\ 0.057,\ 3.532)$ and $(17.190,\ 0.047,\ 31.076)$. The model weights were estimated by minimizing sums of squared residuals: S (left column) and S^+ (right column), see Section B.2 for more details. While the first approach enforces small corrections of f_{prot}, positive values of f_{prot} are favored in the second approach. The following kymographs are displayed: Protrusion component **(A, B)**, APCSF component **(C, D)**, and AAF component **(E, F)**, as well as their proportion on the overall velocity of the contour dynamics **(G, H)**.





Caption on next page, see B.17

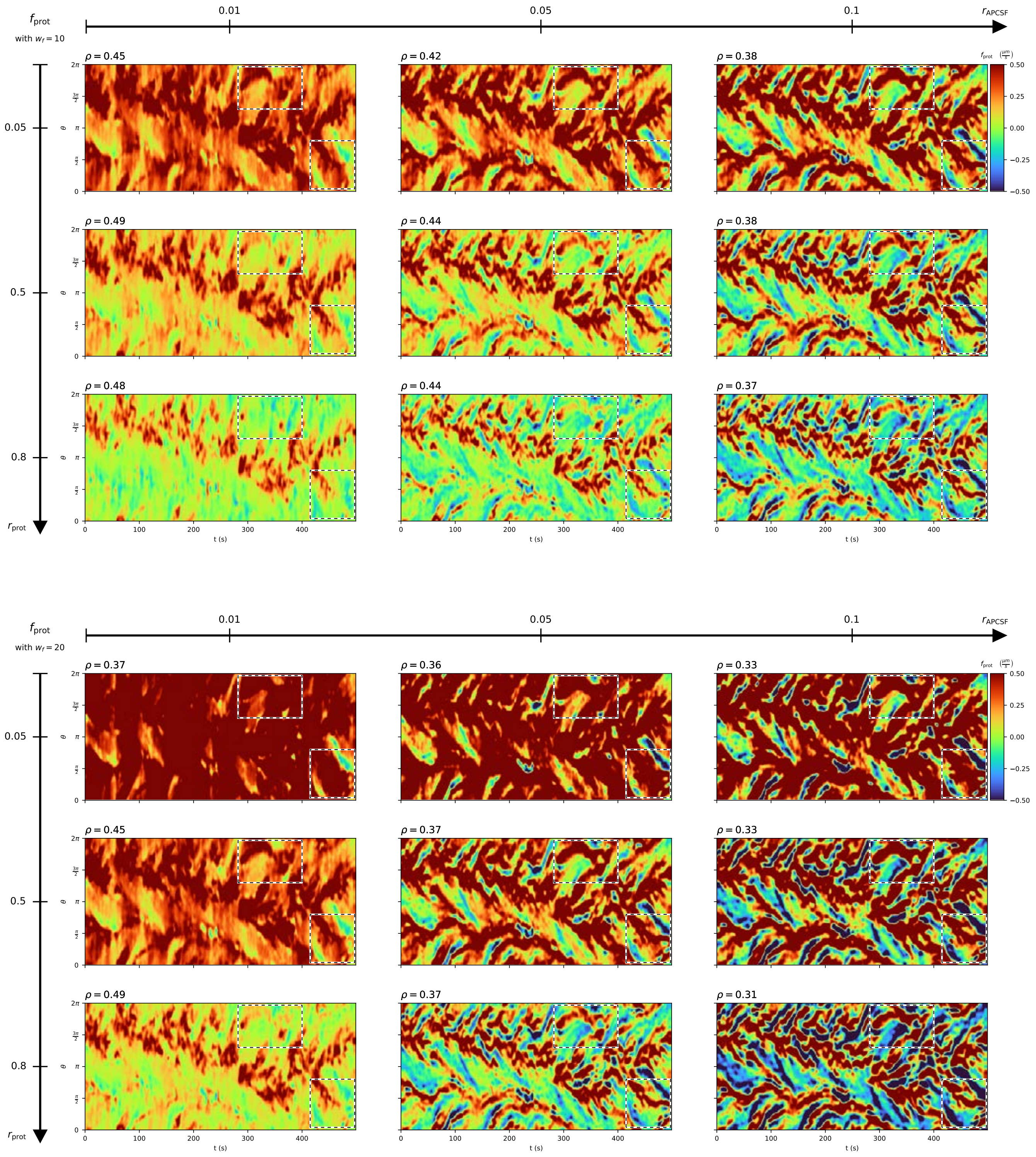


Figure B.17. Protrusion component f_{prot} extracted for varying relative weights $r_{\text{prot}} \in \{0.05, 0.5, 0.8\}$ (vertical axis) and $r_{\text{APCSF}} \in \{0.01, 0.05, 0.1\}$ (horizontal axis) as well as varying overall velocity parameter $w_f \in \{1, 5, 10, 20\}$ (page axis) The correlation coefficient ρ between the protrusion component and the fluorescence intensity kymograph from Fig 5.7D is displayed above each kymograph. Regions of interest are displayed as black and white dashed boxes. The underlying cell track is the same cell track as in Fig 5.7.

B.7 Supporting files

File B.1. Collection of five non-polarized cell tracks based on Poisson point processes as protrusion process and the corresponding kymographs displayed as in Fig 5.3. URL: `https://zenodo.org/record/7243417/files/S4_Fig.pdf` (PDF)

File B.2. Collection of five polarized cell tracks based on Poisson point processes as protrusion process and the corresponding kymographs displayed as in Fig 5.4. URL: `https://zenodo.org/record/7243417/files/S5_Fig.pdf` (PDF)

B.8 Supporting videos

Video B.1. Contour dynamics with corresponding kymographs of a non-polarized cell track based on a Hawkes process as shown in Fig 5.3. The point events generated by the Hawkes process are depicted as white circles. The cell track is displayed at fivefold speed. URL: `https://zenodo.org/record/7243417/files/S1_Vid.mp4` (MP4)

Video B.2. Contour dynamics with corresponding kymographs of a polarized cell track based on a Hawkes process as shown in Fig 5.4. The point events generated by the Hawkes process are depicted as white circles. The cell track is displayed at fivefold speed. URL: `https://zenodo.org/record/7243417/files/S2_Vid.mp4` (MP4)

Video B.3. Contour dynamics for different modifications of the underlying Ornstein-Uhlenbeck process as protrusion process. The cell tracks are displayed at tenfold speed. URL: `https://zenodo.org/record/7243417/files/S3_Vid.mp4` (MP4)

Video B.4. Contour dynamics of non-polarized cell tracks based on a Hawkes process as shown in Fig 5.6. The cell tracks are displayed at tenfold speed. URL: `https://zenodo.org/record/7243417/files/S4_Vid.mp4` (MP4)

Video B.5. Contour dynamics of polarized cell tracks based on a Hawkes process as shown in Fig 5.6. The cell tracks are displayed at tenfold speed. URL: `https://zenodo.org/record/7243417/files/S5_Vid.mp4` (MP4)

Bibliography

[DS1] Daniel Schindler, Ted Moldenhawer, Maike Stange, et al. "Analysis of protrusion dynamics in amoeboid cell motility by means of regularized contour flows". In: *PLoS Computational Biology* 17.8 (2021), e1009268. DOI: 10.1371/journal.pcbi.1009268.

[DS2] Daniel Schindler, Ted Moldenhawer, Carsten Beta, Wilhelm Huisinga, and Matthias Holschneider. "Three-component contour dynamics model to simulate and analyze amoeboid cell motility". In: *arXiv* (2022). DOI: 10.48550/arXiv.2210.12978.

[DS3] Daniel Schindler, Ted Moldenhawer, Lena Lindenmeier, Victor Lesur, and Matthias Holschneider. *AmoePy: A Python-based toolbox for analyzing and simulating amoeboid cell motility*. Zenodo, Software. 2020. DOI: 10.5281/zenodo.3982371.

[DS4] Ted Moldenhawer, Eduardo Moreno, Daniel Schindler, et al. "Spontaneous transitions between amoeboid and keratocyte-like modes of migration". In: *Front. Cell Dev.* 10:898351 (2022). DOI: 10.3389/fcell.2022.898351.

[1] Michael Mihlan, Katharina M. Glaser, Maximilian W. Epple, and Tim Lämmermann. "Neutrophils: Amoeboid Migration and Swarming Dynamics in Tissues". In: *Front. Cell Dev. Biol.* 10:871789 (2022), pp. 1–9. DOI: 10.3389/fcell.2022.871789.

[2] Ronald N. Germain, Ellen A. Robey, and Michael D. Cahalan. "A Decade of Imaging Cellular Motility and Interaction Dynamics in the Immune System". In: *Science* 336.6089 (2012), pp. 1676–1681. DOI: 10.1126/science.1221063.

[3] Tanya J. Shaw and Paul Martin. "Wound repair at a glance". In: *Journal of Cell Science* 122.18 (2009), pp. 3209–3213. DOI: 10.1242/jcs.031187.

[4] John Condeelis, Robert H. Singer, and Jeffrey E. Segall. "THE GREAT ESCAPE: When Cancer Cells Hijack the Genes for Chemotaxis and Motility". In: *Annual Review of Cell and Developmental Biology* 21.1 (2005), pp. 695–718. DOI: 10.1146/annurev.cellbio.21.122303.120306.

[5] Peter Friedl and Stephanie Alexander. "Cancer Invasion and the Microenvironment: Plasticity and Reciprocity". In: *Cell* 147.5 (2011), pp. 992–1009. DOI: 10.1016/j.cell.2011.11.016.

[6] Leonard Bosgraaf and Peter J. M. Van Haastert. “The Ordered Extension of Pseudopodia by Amoeboid Cells in the Absence of External Cues”. In: *PLoS ONE* 4.4 (2009). Ed. by Neil Hotchin, e5253. DOI: 10.1371/journal.pone.0005253.

[7] Yusuke T. Maeda, Junya Inose, Miki Y. Matsuo, Suguru Iwaya, and Masaki Sano. “Ordered Patterns of Cell Shape and Orientational Correlation during Spontaneous Cell Migration”. In: *PLoS ONE* 3.11 (2008). Ed. by Mark Isalan, e3734. DOI: 10.1371/journal.pone.0003734.

[8] Hideko Urushihara. “Cultivation, Spore Production, and Mating”. In: *Dictyostelium discoideum Protocols*. Ed. by Ludwig Eichinger and Francisco Rivero. Humana Press, 2006. Chap. 7, pp. 113–124. DOI: 10.1385/1597451444.

[9] Salvatore Bozarro. “The Model Organism Dictyostelium discoideum”. In: *Dictyostelium discoideum Protocols*. Ed. by Ludwig Eichinger and Francisco Rivero. Humana Press, 2013. Chap. 2, pp. 17–37. DOI: 10.1007/978-1-62703-302-2.

[10] David Selmeczi, Stephan Mosler, Peter H. Hagedorn, Niels B. Larsen, and Henrik Flyvbjerg. “Cell Motility as Persistent Random Motion: Theories from Experiments”. In: *Biophysical Journal* 89.2 (2005), pp. 912–931. DOI: 10.1529/biophysj.105.061150.

[11] H. U. Bödeker, C. Beta, T. D. Frank, and E. Bodenschatz. “Quantitative analysis of random ameboid motion”. In: *EPL (Europhysics Letters)* 90.2 (2010), p. 28005. DOI: 10.1209/0295-5075/90/28005.

[12] Natallia Makarava, Stephan Menz, Matthias Theves, et al. “Quantifying the degree of persistence in random amoeboid motion based on the Hurst exponent of fractional Brownian motion”. In: *Physical Review E* 90.4 (2014), p. 042703. DOI: 10.1103/PhysRevE.90.042703.

[13] A. Dreher, I. S. Aranson, and K. Kruse. “Spiral actin-polymerization waves can generate amoeboidal cell crawling”. In: *New Journal of Physics* 16.5 (2014), p. 055007. DOI: 10.1088/1367-2630/16/5/055007.

[14] Sergio Alonso, Maike Stange, and Carsten Beta. “Modeling random crawling, membrane deformation and intracellular polarity of motile amoeboid cells”. In: *PLOS ONE* 13.8 (2018). Ed. by Thierry Soldati, e0201977. DOI: 10.1371/journal.pone.0201977.

[15] Tommy Heck, Diego A. Vargas, Bart Smeets, et al. “The role of actin protrusion dynamics in cell migration through a degradable viscoelastic extracellular matrix: Insights from a computational model”. In: *PLOS Computational Biology* 16.1 (2020). Ed. by Roeland M.H. Merks, e1007250. DOI: 10.1371/journal.pcbi.1007250.

[16] Cheng-Jin Du, Phillip T. Hawkins, Len R. Stephens, and Till Bretschneider. "3D time series analysis of cell shape using Laplacian approaches". In: *BMC Bioinformatics* 14.1 (2013), p. 296. DOI: 10.1186/1471-2105-14-296.

[17] David R. Soll, Edward Voss, Olof Johnson, and Deborah Wessels. "Three-dimensional reconstruction and motion analysis of living, crawling cells". In: *Scanning* 22.4 (2006), pp. 249–257. DOI: 10.1002/sca.4950220404.

[18] Leonard Bosgraaf and Peter J. M. Van Haastert. "Navigation of Chemotactic Cells by Parallel Signaling to Pseudopod Persistence and Orientation". In: *PLoS ONE* 4.8 (2009). Ed. by Terry Means, e6842. DOI: 10.1371/journal.pone.0006842.

[19] Meghan K. Driscoll, Colin McCann, Rael Kopace, et al. "Cell Shape Dynamics: From Waves to Migration". In: *PLoS Computational Biology* 8.3 (2012). Ed. by Jason M. Haugh, e1002392. DOI: 10.1371/journal.pcbi.1002392.

[20] Natalie Andrew and Robert H. Insall. "Chemotaxis in shallow gradients is mediated independently of PtdIns 3-kinase by biased choices between random protrusions". In: *Nature Cell Biology* 9.2 (2007), pp. 193–200. DOI: 10.1038/ncb1536.

[21] Erin L. Barnhart, Kun-Chun Lee, Kinneret Keren, Alex Mogilner, and Julie A. Theriot. "An Adhesion-Dependent Switch between Mechanisms That Determine Motile Cell Shape". In: *PLoS Biology* 9.5 (2011). Ed. by Jonathan B. Alberts, e1001059. DOI: 10.1371/journal.pbio.1001059.

[22] Javier Satulovsky, Roger Lui, and Yu-li Wang. "Exploring the Control Circuit of Cell Migration by Mathematical Modeling". In: *Biophysical Journal* 94.9 (2008), pp. 3671–3683. DOI: 10.1529/biophysj.107.117002.

[23] Leonard Bosgraaf and Peter J.M. Van Haastert. "Quimp3, an automated pseudopod-tracking algorithm". In: *Cell Adhesion & Migration* 4.1 (2010), pp. 46–55. DOI: 10.4161/cam.4.1.9953.

[24] Meghan K. Driscoll, John T. Fourkas, and Wolfgang Losert. "Local and global measures of shape dynamics". In: *Physical Biology* 8.5 (2011), p. 055001. DOI: 10.1088/1478-3975/8/5/055001.

[25] Robert M. Cooper, Ned S. Wingreen, and Edward C. Cox. "An Excitable Cortex and Memory Model Successfully Predicts New Pseudopod Dynamics". In: *PLoS ONE* 7.3 (2012). Ed. by Thierry Soldati, e33528. DOI: 10.1371/journal.pone.0033528.

[26] Yuan Xiong, Cathryn Kabacoff, Jonathan Franca-Koh, et al. "Automated characterization of cell shape changes during amoeboid motility by skeletonization". In: *BMC Systems Biology* 4.1 (2010), p. 33. DOI: 10.1186/1752-0509-4-33.

[27] Charles W. Wolgemuth and Mark Zajac. “The moving boundary node method: A level set-based, finite volume algorithm with applications to cell motility”. In: *Journal of Computational Physics* 229.19 (2010), pp. 7287–7308. DOI: 10.1016/j.jcp.2010.06.014.

[28] R.A. Tyson, D.B.A. Epstein, K.I. Anderson, and T. Bretschneider. “High Resolution Tracking of Cell Membrane Dynamics in Moving Cells: an Electrifying Approach”. In: *Mathematical Modelling of Natural Phenomena* 5.1 (2010), pp. 34–55. DOI: 10.1051/mmnp/20105102.

[29] M. Machacek and G. Danuser. “Morphodynamic Profiling of Protrusion Phenotypes”. In: *Biophysical Journal* 90.4 (2006), pp. 1439–1452. DOI: 10.1529/biophysj.105.070383.

[30] Hiroaki Takagi, Masayuki J. Sato, Toshio Yanagida, and Masahiro Ueda. “Functional Analysis of Spontaneous Cell Movement under Different Physiological Conditions”. In: *PLoS ONE* 3.7 (2008). Ed. by Eshel Ben-Jacob, e2648. DOI: 10.1371/journal.pone.0002648.

[31] Gabriel Amselem, Matthias Theves, Albert Bae, Eberhard Bodenschatz, and Carsten Beta. “A Stochastic Description of Dictyostelium Chemotaxis”. In: *PLoS ONE* 7.5 (2012). Ed. by Adrian John Harwood, e37213. DOI: 10.1371/journal.pone.0037213.

[32] Liang Li, Edward C. Cox, and Henrik Flyvbjerg. “‘Dicty dynamics’: Dictyostelium motility as persistent random motion”. In: *Physical Biology* 8.4 (2011), p. 046006. DOI: 10.1088/1478-3975/8/4/046006.

[33] Zahra Eidi. “Discrete Modeling of Amoeboid Locomotion and Chemotaxis in Dictyostelium discoideum by Tracking Pseudopodium Growth Direction”. In: *Scientific Reports* 7.1 (2017), p. 12675. DOI: 10.1038/s41598-017-12656-1.

[34] Pablo A. Iglesias and Peter N. Devreotes. “Biased excitable networks: how cells direct motion in response to gradients”. In: *Current Opinion in Cell Biology* 24.2 (2012), pp. 245–253. DOI: 10.1016/j.ceb.2011.11.009.

[35] Changji Shi, Chuan-Hsiang Huang, Peter N. Devreotes, and Pablo A. Iglesias. “Interaction of Motility, Directional Sensing, and Polarity Modules Recreates the Behaviors of Chemotaxing Cells”. In: *PLoS Computational Biology* 9.7 (2013). Ed. by Jason M. Haugh, e1003122. DOI: 10.1371/journal.pcbi.1003122.

[36] Matthew P. Neilson, Douwe M. Veltman, Peter J. M. van Haastert, et al. “Chemotaxis: A Feedback-Based Computational Model Robustly Predicts Multiple Aspects of Real Cell Behaviour”. In: *PLoS Biology* 9.5 (2011). Ed. by Thomas Pollard, e1000618. DOI: 10.1371/journal.pbio.1000618.

[37] M. P. Neilson, J. A. Mackenzie, S. D. Webb, and R. H. Insall. "Modeling Cell Movement and Chemotaxis Using Pseudopod-Based Feedback". In: *SIAM Journal on Scientific Computing* 33.3 (2011), pp. 1035–1057. DOI: 10.1137/100788938.

[38] Tatsuo Shibata, Masatoshi Nishikawa, Satomi Matsuoka, and Masahiro Ueda. "Modeling the self-organized phosphatidylinositol lipids signaling system in chemotactic cells based on quantitative image analysis". In: *Journal of Cell Science* 125.21 (2012), pp. 5138–5150. DOI: 10.1242/jcs.108373.

[39] Tatsuo Shibata, Masatoshi Nishikawa, Satomi Matsuoka, and Masahiro Ueda. "Intracellular Encoding of Spatiotemporal Guidance Cues in a Self-Organizing Signaling System for Chemotaxis in Dictyostelium Cells". In: *Biophysical Journal* 105.9 (2013), pp. 2199–2209. DOI: 10.1016/j.bpj.2013.09.024.

[40] Daisuke Taniguchi, Shuji Ishihara, Takehiko Oonuki, et al. "Phase geometries of two-dimensional excitable waves govern self-organized morphodynamics of amoeboid cells". In: *Proceedings of the National Academy of Sciences* 110.13 (2013), pp. 5016–5021. DOI: 10.1073/pnas.1218025110.

[41] Zahra Eidi, Farshid Mohammad-Rafiee, Mohammad Khorrami, and Azam Gholami. "Modelling of Dictyostelium discoideum movement in a linear gradient of chemoattractant". In: *Soft Matter* 13.44 (2017), pp. 8209–8222. DOI: 10.1039/C7SM01568B.

[42] M Gracheva. "A continuum model of motility in ameboid cells". In: *Bulletin of Mathematical Biology* 66.1 (2004), pp. 167–193. DOI: 10.1016/j.bulm.2003.08.007.

[43] Mathias Buenemann, Herbert Levine, Wouter-Jan Rappel, and Leonard M. Sander. "The Role of Cell Contraction and Adhesion in Dictyostelium Motility". In: *Biophysical Journal* 99.1 (2010), pp. 50–58. DOI: 10.1016/j.bpj.2010.03.057.

[44] Francisco Merino-Casallo, Maria J. Gomez-Benito, Yago Juste-Lanas, Ruben Martinez-Cantin, and Jose M. Garcia-Aznar. "Integration of in vitro and in silico Models Using Bayesian Optimization With an Application to Stochastic Modeling of Mesenchymal 3D Cell Migration". In: *Front. Physiol.* 9:1246 (2018). DOI: 10.3389/fphys.2018.01246.

[45] Amy L. Bauer, Trachette L. Jackson, and Yi Jiang. "Topography of Extracellular Matrix Mediates Vascular Morphogenesis and Migration Speeds in Angiogenesis". In: *PLoS Computational Biology* 5.7 (2009). Ed. by András Czirók, e1000445. DOI: 10.1371/journal.pcbi.1000445.

[46] Carlos Borau, Taeyoon Kim, Tamara Bidone, José Manuel García-Aznar, and Roger D. Kamm. "Dynamic Mechanisms of Cell Rigidity Sensing: Insights from a Computational Model of Actomyosin Networks". In: *PLoS ONE* 7.11 (2012). Ed. by Wilbur Lam, e49174. DOI: 10.1371/journal.pone.0049174.

[47] Josephine T. Daub and Roeland M. H. Merks. “A Cell-Based Model of Extracellular-Matrix-Guided Endothelial Cell Migration During Angiogenesis”. In: *Bulletin of Mathematical Biology* 75.8 (2013), pp. 1377–1399. DOI: 10.1007/s11538-013-9826-5.

[48] Victoria Stodden, David Donoho, Sergey Fomel, et al. “Reproducible Research”. In: *Computing in Science & Engineering* 12.5 (2010), pp. 8–13. DOI: 10.1109/MCSE.2010.113.

[49] Monya Baker. “1,500 scientists lift the lid on reproducibility”. In: *Nature* 533.7604 (2016), pp. 452–454. DOI: 10.1038/533452a.

[50] Marcus R. Munafò, Brian A. Nosek, Dorothy V. M. Bishop, et al. “A manifesto for reproducible science”. In: *Nature Human Behaviour* 1.1 (2017), p. 0021. DOI: 10.1038/s41562-016-0021.

[51] Lorena A. Barba. “Terminologies for Reproducible Research”. In: *arXiv* (2018). DOI: 10.48550/arXiv.1802.03311.

[52] Daniele Fanelli. “Is science really facing a reproducibility crisis, and do we need it to?” In: *Proceedings of the National Academy of Sciences* 115.11 (2018), pp. 2628–2631. DOI: 10.1073/pnas.1708272114.

[53] Erin C. McKiernan, Philip E. Bourne, C. Titus Brown, et al. “Point of View: How open science helps researchers succeed”. In: *eLife* 5 (2016), e16800. DOI: 10.7554/eLife.16800.

[54] Jean-Claude Burgelman, Corina Pascu, Katarzyna Szkuta, et al. “Open Science, Open Data, and Open Scholarship: European Policies to Make Science Fit for the Twenty-First Century”. In: *Front. Big Data* 2:43 (2019), pp. 1–6. DOI: 10.3389/fdata.2019.00043.

[55] Rachael V. Gallagher, Daniel S. Falster, Brian S. Maitner, et al. “Open Science principles for accelerating trait-based science across the Tree of Life”. In: *Nature Ecology & Evolution* 4.3 (2020), pp. 294–303. DOI: 10.1038/s41559-020-1109-6.

[56] Christian Heise and Joshua M. Pearce. “From Open Access to Open Science: The Path From Scientific Reality to Open Scientific Communication”. In: *SAGE Open* 10.2 (2020). DOI: 10.1177/2158244020915900.

[57] Christina Schilde and Pauline Schaap. “The Amoebozoa”. In: *Dictyostelium discoideum Protocols*. Ed. by Ludwig Eichinger and Francisco Rivero. Humana Press, 2013. Chap. 1, pp. 1–15. DOI: 10.1007/978-1-62703-302-2.

[58] Sabina Leonelli. “Model Organism”. In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky, Olaf Wolkenhauer, Kwang-Hyun Cho, and Hiroki Yokota. New York, NY: Springer New York, 2013, pp. 1398–1401. DOI: 10.1007/978-1-4419-9863-7_76.

[59] Andrew R. Swanson, Eduardo M. Vadell, and James C. Cavender. “Global distribution of forest soil dictyostelids”. In: *Journal of Biogeography* 26.1 (1999), pp. 133–148. DOI: 10.1046/j.1365-2699.1999.00250.x.

[60] David R. Waddell. “A predatory slime mould”. In: *Nature* 298.5873 (1982), pp. 464–466. DOI: 10.1038/298464a0.

[61] Yasuo Maeda and Tomokazu Kawamoto. “Pinocytosis in Dictyostelium discoideum cells”. In: *Experimental Cell Research* 164.2 (1986), pp. 516–526. DOI: 10.1016/0014-4827(86)90049-2.

[62] Unha Baik. *Einfluss von CbfA auf Wachstum und Entwicklung in Dictyostelium discoideum*. PhD Thesis. Goethe-Universität Frankfurt am Main, 2004, pp. 1-117. URL: http://d-nb.info/974551120.

[63] Ingo W. M. Jennes. *Differentielle Expressionsanalyse CbfA-regulierter Proteine während des Übergangs vom Wachstum zur Entwicklung in Dictyostelium discoideum*. PhD Thesis. Goethe-Universität Frankfurt am Main, 2006, pp. 1-130. URL: http://d-nb.info/982025483.

[64] Christina Schilde, Pauline Schaap, Salvatore Bozarro, et al. *Dictyostelium discoideum Protocols*. Ed. by Ludwig Eichinger and Francisco Rivero. Vol. 983. Methods in Molecular Biology. Totowa, NJ: Humana Press, 2013, pp. 1–479. DOI: 10.1007/978-1-62703-302-2.

[65] S. Blair Hedges. “The origin and evolution of model organisms”. In: *Nature Reviews Genetics* 3.11 (2002), pp. 838–849. DOI: 10.1038/nrg929.

[66] Dennis Bray. *Cell Movements: From Molecules To Motility*. Ed. by Matthew Day. 2nd. New York: Garland Science, 2000, p. 386. DOI: 10.4324/9780203833582.

[67] J. Victor Small, Klemens Rottner, Shiro Suetsugu, et al. *Actin-based Motility*. Ed. by Marie-France Carlier. Dordrecht: Springer Netherlands, 2010, pp. 1–435. DOI: 10.1007/978-90-481-9301-1.

[68] Peter J. M. Van Haastert. “How Cells Use Pseudopods for Persistent Movement and Navigation”. In: *Science Signaling* 4.159 (2011), pp. 6–9. DOI: 10.1126/scisignal.2001708.

[69] Douglas A. Lauffenburger and Alan F. Horwitz. “Cell Migration: A Physically Integrated Molecular Process”. In: *Cell* 84.3 (1996), pp. 359–369. DOI: 10.1016/S0092-8674(00)81280-5.

[70] Laurent Blanchoin, Rajaa Boujemaa-Paterski, Cécile Sykes, and Julie Plastino. “Actin Dynamics, Architecture, and Mechanics in Cell Motility”. In: *Physiological Reviews* 94.1 (2014), pp. 235–263. DOI: 10.1152/physrev.00018.2013.

[71] Anne J. Ridley, Martin A. Schwartz, Keith Burridge, et al. “Cell Migration: Integrating Signals from Front to Back”. In: *Science* 302.5651 (2003), pp. 1704–1709. DOI: 10.1126/science.1092053.

[72] Guangwu Xu and Yufang Shi. “Apoptosis signaling pathways and lymphocyte homeostasis”. In: *Cell Research* 17.9 (2007), pp. 759–771. DOI: 10.1038/cr.2007.52.

[73] Peter J. M. van Haastert and Peter N. Devreotes. “Chemotaxis: signalling the way forward”. In: *Nature Reviews Molecular Cell Biology* 5.8 (2004), pp. 626–634. DOI: 10.1038/nrm1435.

[74] Ryan J. Petrie and Kenneth M. Yamada. “At the leading edge of three-dimensional cell migration”. In: *Journal of Cell Science* 125.24 (2012), pp. 5917–5926. DOI: 10.1242/jcs.093732.

[75] Eduardo Moreno, Sven Flemming, Francesc Font, et al. “Modeling cell crawling strategies with a bistable model: From amoeboid to fan-shaped cell motion”. In: *Physica D: Nonlinear Phenomena* 412 (2020), p. 132591. DOI: 10.1016/j.physd.2020.132591.

[76] Guillaume Jacquemet, Hellyeh Hamidi, and Johanna Ivaska. “Filopodia in cell adhesion, 3D migration and cancer cell invasion”. In: *Current Opinion in Cell Biology* 36 (2015), pp. 23–31. DOI: 10.1016/j.ceb.2015.06.007.

[77] Wallace F. Marshall. “Axopodia and the cellular “arms” race”. In: *Cytoskeleton* 77.11 (2020), pp. 483–484. DOI: 10.1002/cm.21642.

[78] Samuel S. Bowser. “Reticulopodia: structural and behavioral basis for the suprageneric placement of Granuloreticulosan protists”. In: *The Journal of Foraminiferal Research* 32.4 (2002), pp. 440–447. DOI: 10.2113/0320440.

[79] Sharon Collier, Peggy Paschke, Robert R. Kay, and Till Bretschneider. “Image based modeling of bleb site selection”. In: *Scientific Reports* 7.1 (2017), p. 6692. DOI: 10.1038/s41598-017-06875-9.

[80] Evgeny Zatulovskiy, Richard Tyson, Till Bretschneider, and Robert R. Kay. “Bleb-driven chemotaxis of Dictyostelium cells”. In: *Journal of Cell Biology* 204.6 (2014), pp. 1027–1044. DOI: 10.1083/jcb.201306147.

[81] Revathi Ananthakrishnan and Allen Ehrlicher. “The Forces Behind Cell Movement”. In: *International Journal of Biological Sciences* 3.5 (2007), pp. 303–317. DOI: 10.7150/ijbs.3.303.

[82] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. Ed. by Thomas Dietterich. Adaptive Computation and Machine Learning. MIT Press, 2005. DOI: 10.7551/mitpress/3206.001.0001.

[83] Carl Edward Rasmussen. “Gaussian Processes in machine learning”. In: *Advanced Lectures on Machine Learning*. Ed. by Olivier Bousquet, Ulrike Luxburg, and Gunnar Rätsch. 1st ed. Vol. 3176. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2003, pp. 63–71. DOI: 10.1007/978-3-540-28650-9_4.

[84] C. K. I. Williams and C. E. Rasmussen. *Gaussian processes for regression*. In: Advances in Neural Information Processing Systems. Ed. by D. Touretzky, M. C. Mozer, and M. Hasselmo. Vol. 8. MIT Press, 1996, pp. 514-520. URL: http://eprints.aston.ac.uk/651/.

[85] C. K. I. Williams. “Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond”. In: *Learning in Graphical Models*. Dordrecht: Springer Netherlands, 1998, pp. 599–621. DOI: 10.1007/978-94-011-5014-9_23.

[86] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O’Neil. “Fast Direct Methods for Gaussian Processes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2016), pp. 252–265. DOI: 10.1109/TPAMI.2015.2448083.

[87] Subhasish Basak, Sébastien Petit, Julien Bect, and Emmanuel Vazquez. “Numerical Issues in Maximum Likelihood Parameter Estimation for Gaussian Process Interpolation”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 13164 LNCS. 2022, pp. 116–131. DOI: 10.1007/978-3-030-95470-3_9.

[88] Toni Karvonen, George Wynne, Filip Tronarp, Chris Oates, and Simo Sarkka. “Maximum likelihood estimation and uncertainty quantification for gaussian process approximation of deterministic functions”. In: *SIAM-ASA Journal on Uncertainty Quantification* 8.3 (2020), pp. 926–958. DOI: 10.1137/20M1315968.

[89] Toni Karvonen and Chris J. Oates. “Maximum Likelihood Estimation in Gaussian Process Regression is Ill-Posed”. In: *arXiv* (2022). DOI: 10.48550/arXiv.2203.09179.

[90] S. Sundararajan and S. S. Keerthi. “Predictive approaches for choosing hyperparameters in gaussian processes”. In: *Neural Computation* 13.5 (2001), pp. 1103–1118. DOI: 10.1162/08997660151134343.

[91] François Bachoc, Agnès Lagnoux, and Thi Mong Ngoc Nguyen. “Cross-validation estimation of covariance parameters under fixed-domain asymptotics”. In: *Journal of Multivariate Analysis* 160 (2017), pp. 42–67. DOI: 10.1016/j.jmva.2017.06.003.

[92] A. Stéphanou, M. A. J. Chaplain, and P. Tracqui. “A mathematical model for the dynamics of large membrane deformations of isolated fibroblasts”. In: *Bulletin of Mathematical Biology* 66.5 (2004), pp. 1119–1154. DOI: 10.1016/j.bulm.2003.11.004.

[93] Satyanad Kichenassamy, Arun Kumar, Peter Olver, Allen Tannenbaum, and Anthony Yezzi. “Gradient flows and geometric active contour models”. In: *Proceedings of IEEE International Conference on Computer Vision*. IEEE Comput. Soc. Press, 1995, pp. 810–815. DOI: 10.1109/ICCV.1995.466855.

[94] Kaleem Siddiqi, Y.B. Lauziere, Allen Tannenbaum, and S.W. Zucker. “Area and length minimizing flows for shape segmentation”. In: *IEEE Transactions on Image Processing* 7.3 (1998), pp. 433–443. DOI: 10.1109/83.661193.

[95] Italo Capuzzo Dolcetta, Stefano Finzi Vita, and Riccardo March. “Area-preserving curve-shortening flows: from phase separation to image processing”. In: *Interfaces and Free Boundaries* 4.4 (2002), pp. 325–343. DOI: 10.4171/IFB/64.

[96] Elena Mäder-Baumdicker. “The area preserving curve shortening flow with Neumann free boundary conditions”. In: *Geometric Flows* 1.1 (2015), pp. 1–57. DOI: 10.1515/geofl-2015-0004.

[97] Elena Mäder-Baumdicker. “Singularities of the area preserving curve shortening flow with a free boundary condition”. In: *Mathematische Annalen* 371.3-4 (2018), pp. 1429–1448. DOI: 10.1007/s00208-017-1637-9.

[98] Patrick J. Laub, Young Lee, and Thomas Taimre. *The Elements of Hawkes Processes*. 1st ed. Cham: Springer International Publishing, 2021, p. 133. DOI: 10.1007/978-3-030-84639-8.

[99] Patrick J. Laub, Thomas Taimre, and Philip K. Pollett. “Hawkes Processes”. In: *arXiv* (2015). DOI: 10.48550/arXiv.1507.02822.

[100] Alex Reinhart. “A Review of Self-Exciting Spatio-Temporal Point Processes and Their Applications”. In: *Statistical Science* 33.3 (2018), pp. 299–318. DOI: 10.1214/17-STS629.

[101] Grigorios A. Pavliotis. *Stochastic Processes and Applications*. 1st ed. Vol. 60. Texts in Applied Mathematics. New York, NY: Springer New York, 2014, p. 339. DOI: 10.1007/978-1-4939-1323-7.

[102] ALAN G. HAWKES. “Spectra of some self-exciting and mutually exciting point processes”. In: *Biometrika* 58.1 (1971), pp. 83–90. DOI: 10.1093/biomet/58.1.83.

[103] Alan G. Hawkes. “Point Spectra of Some Mutually Exciting Point Processes”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 33.3 (1971), pp. 438–443. DOI: 10.1111/j.2517-6161.1971.tb01530.x.

[104] Alan G. Hawkes and David Oakes. “A cluster process representation of a self-exciting process”. In: *Journal of Applied Probability* 11.3 (1974), pp. 493–503. DOI: 10.2307/3212693.

[105] G. E. Uhlenbeck and L. S. Ornstein. “On the Theory of the Brownian Motion”. In: *Physical Review* 36.5 (1930), pp. 823–841. DOI: 10.1103/PhysRev.36.823.

[106] Felix Otto. “The geometry of dissipative evolution equations: The porous medium equation”. In: *Communications in Partial Differential Equations* 26.1-2 (2001), pp. 101–174. DOI: 10.1081/PDE-100002243.

[107] Rachel Fink. *CIL:7792. Dataset. Fundulus heteroclitus, early embryonic cell, deep cells*. 2010. DOI: 10.7295/W9CIL7792.

[108] Rachel Fink and Pat Wadsworth. *CIL:35204. Dataset. Fundulus heteroclitus, deep cell*. 2011. DOI: 10.7295/W9CIL35204.

[109] Rachel Fink and Pat Wadsworth. *CIL:35205. Dataset. Fundulus heteroclitus, deep cell*. 2011. DOI: 10.7295/W9CIL35205.

[110] John Murray, Holly Vawter-Hugart, Edward Voss, and David R. Soll. "Three-dimensional motility cycle in leukocytes". In: *Cell Motility and the Cytoskeleton* 22.3 (1992), pp. 211–223. DOI: 10.1002/cm.970220308.

[111] Deborah Wessels, Edward Voss, Nick Von Bergen, et al. "A computer-assisted system for reconstructing and interpreting the dynamic three-dimensional relationships of the outer surface, nucleus and pseudopods of crawling cells". In: *Cell Motility and the Cytoskeleton* 41.3 (1998), pp. 225–246. DOI: 10.1002/(SICI)1097-0169(1998)41:3<225::AID-CM4>3.0.CO;2-I.

[112] G.A. Dunn and D. Zicha. "Dynamics of fibroblast spreading". In: *Journal of Cell Science* 108.3 (1995), pp. 1239–1249. DOI: 10.1242/jcs.108.3.1239.

[113] Klemens Rottner, Barbara Behrendt, J.Victor Small, and Jurgen Wehland. "VASP dynamics during lamellipodia protrusion". In: *Nature Cell Biology* 1.5 (1999), pp. 321–322. DOI: 10.1038/13040.

[114] Boris Hinz, Wolfgang Alt, Christa Johnen, Volker Herzog, and Hans-Wilhelm Kaiser. "Quantifying Lamella Dynamics of Cultured Cells by SACED, a New Computer-Assisted Motion Analysis". In: *Experimental Cell Research* 251.1 (1999), pp. 234–243. DOI: 10.1006/excr.1999.4541.

[115] Go Totsukawa, Yue Wu, Yasuharu Sasaki, et al. "Distinct roles of MLCK and ROCK in the regulation of membrane protrusions and focal adhesion dynamics during cell migration of fibroblasts". In: *Journal of Cell Biology* 164.3 (2004), pp. 427–439. DOI: 10.1083/jcb.200306172.

[116] Dirk Dormann, Thorsten Libotte, Cornelis J. Weijer, and Till Bretschneider. "Simultaneous quantification of cell motility and protein-membrane-association using active contours". In: *Cell Motility and the Cytoskeleton* 52.4 (2002), pp. 221–230. DOI: 10.1002/cm.10048.

[117] Leonard Bosgraaf, Peter J.M. van Haastert, and Till Bretschneider. "Analysis of cell movement by simultaneous quantification of local membrane displacement and fluorescent intensities using Quimp2". In: *Cell Motility and the Cytoskeleton* 66.3 (2009), pp. 156–165. DOI: 10.1002/cm.20338.

[118] Carole A. Parent and Peter N. Devreotes. "A Cell's Sense of Direction". In: *Science* 284.5415 (1999), pp. 765–770. DOI: 10.1126/science.284.5415.765.

[119] Yuan Xiong, Chuan-Hsiang Huang, Pablo A. Iglesias, and Peter N. Devreotes. "Cells navigate with a local-excitation, global-inhibition-biased excitable network". In: *Proceedings of the National Academy of Sciences* 107.40 (2010), pp. 17079–17086. DOI: 10.1073/pnas.1011271107.

[120] Brian A. Camley, Yanxiang Zhao, Bo Li, Herbert Levine, and Wouter-Jan Rappel. “Crawling and turning in a minimal reaction-diffusion cell motility model: Coupling cell shape and biochemistry”. In: *Physical Review E* 95.1 (2017), p. 012401. DOI: 10.1103/PhysRevE.95.012401.

[121] Carsten Beta and Karsten Kruse. “Intracellular Oscillations and Waves”. In: *Annual Review of Condensed Matter Physics* 8.1 (2017), pp. 239–264. DOI: 10.1146/annurev-conmatphys-031016-025210.

[122] A.D. Shenderov and M.P. Sheetz. “Inversely correlated cycles in speed and turning in an ameba: an oscillatory model of cell locomotion”. In: *Biophysical Journal* 72.5 (1997), pp. 2382–2389. DOI: 10.1016/S0006-3495(97)78883-0.

[123] Brian A. Camley, Yanxiang Zhao, Bo Li, Herbert Levine, and Wouter-Jan Rappel. “Periodic Migration in a Physical Model of Cells on Micropatterns”. In: *Physical Review Letters* 111.15 (2013), p. 158102. DOI: 10.1103/PhysRevLett.111.158102.

[124] Adrian Moure and Hector Gomez. “Computational model for amoeboid motion: Coupling membrane and cytosol dynamics”. In: *Physical Review E* 94.4 (2016), p. 042423. DOI: 10.1103/PhysRevE.94.042423.

[125] Danying Shao, Wouter-Jan Rappel, and Herbert Levine. “Computational Model for Cell Morphodynamics”. In: *Physical Review Letters* 105.10 (2010), p. 108104. DOI: 10.1103/PhysRevLett.105.108104.

[126] Danying Shao, Herbert Levine, and Wouter-Jan Rappel. “Coupling actin flow, adhesion, and morphology in a computational cell motility model”. In: *Proceedings of the National Academy of Sciences* 109.18 (2012), pp. 6851–6856. DOI: 10.1073/pnas.1203252109.

[127] Jakob Löber, Falko Ziebert, and Igor S. Aranson. “Modeling crawling cell movement on soft engineered substrates”. In: *Soft Matter* 10.9 (2014), pp. 1365–1373. DOI: 10.1039/C3SM51597D.

[128] Falko Ziebert and Igor S. Aranson. “Computational approaches to substrate-based cell motility”. In: *npj Computational Materials* 2.1 (2016), p. 16019. DOI: 10.1038/npjcompumats.2016.19.

[129] Sara Najem and Martin Grant. “Phase-field approach to chemotactic driving of neutrophil morphodynamics”. In: *Physical Review E* 88.3 (2013), p. 034702. DOI: 10.1103/PhysRevE.88.034702.

[130] Charles M. Elliott, Björn Stinner, and Chandrasekhar Venkataraman. “Modelling cell motility and chemotaxis with evolving surface finite elements”. In: *Journal of the Royal Society Interface* 9.76 (2012), pp. 3027–3044. DOI: 10.1098/rsif.2012.0276.

[131] A. C. Callan-Jones and R. Voituriez. “Active gel model of amoeboid cell motility”. In: *New Journal of Physics* 15.2 (2013), p. 025022. DOI: 10.1088/1367-2630/15/2/025022.

[132] Ido Lavi, Nicolas Meunier, Raphael Voituriez, and Jaume Casademunt. "Motility and morphodynamics of confined cells". In: *Physical Review E* 101.2 (2020), p. 022404. DOI: 10.1103/PhysRevE.101.022404.

[133] Jie Zhu and Alex Mogilner. "Comparison of cell migration mechanical strategies in three-dimensional matrices: a computational study". In: *Interface Focus* 6.5 (2016), p. 20160040. DOI: 10.1098/rsfs.2016.0040.

[134] Hao Wu, M. Thiébaud, W.-F. Hu, et al. "Amoeboid motion in confined geometry". In: *Physical Review E* 92.5 (2015), p. 050701. DOI: 10.1103/PhysRevE.92.050701.

[135] Eric J. Campbell and Prosenjit Bagchi. "A computational model of amoeboid cell swimming". In: *Physics of Fluids* 29.10 (2017), p. 101902. DOI: 10.1063/1.4990543.

[136] Chuan-Hsiang Huang, Ming Tang, Changji Shi, Pablo A. Iglesias, and Peter N. Devreotes. "An excitable signal integrator couples to an idling cytoskeletal oscillator to drive cell migration". In: *Nature Cell Biology* 15.11 (2013), pp. 1307–1316. DOI: 10.1038/ncb2859.

[137] Ming Tang, Mingjie Wang, Changji Shi, et al. "Evolutionarily conserved coupling of adaptive and excitable networks mediates eukaryotic chemotaxis". In: *Nature Communications* 5.1 (2014), p. 5175. DOI: 10.1038/ncomms6175.

[138] Peter J.M. van Haastert. "A Stochastic Model for Chemotaxis Based on the Ordered Extension of Pseudopods". In: *Biophysical Journal* 99.10 (2010), pp. 3345–3354. DOI: 10.1016/j.bpj.2010.09.042.

[139] G. MacDonald, J.A. Mackenzie, M. Nolan, and R.H. Insall. "A computational method for the coupled solution of reaction–diffusion equations on evolving domains and manifolds: Application to a model of cell migration and chemotaxis". In: *Journal of Computational Physics* 309 (2016), pp. 207–226. DOI: 10.1016/j.jcp.2015.12.038.

[140] J. A. Mackenzie, M. Nolan, and R. H. Insall. "Local modulation of chemoattractant concentrations by single cells: dissection using a bulk-surface computational model". In: *Interface Focus* 6.5 (2016), p. 20160036. DOI: 10.1098/rsfs.2016.0036.

[141] J. A. Mackenzie, M. Nolan, C. F. Rowlatt, and R. H. Insall. "An Adaptive Moving Mesh Method for Forced Curve Shortening Flow". In: *SIAM Journal on Scientific Computing* 41.2 (2019), A1170–A1200. DOI: 10.1137/18M1211969.

[142] John Mackenzie, Christopher Rowlatt, and Robert Insall. "A Conservative Finite Element ALE Scheme for Mass-Conservative Reaction-Diffusion Equations on Evolving Two-Dimensional Domains". In: *SIAM Journal on Scientific Computing* 43.1 (2021), B132–B166. DOI: 10.1137/19M1298585.

[143] William R. Holmes and Leah Edelstein-Keshet. “A Comparison of Computational Models for Eukaryotic Cell Shape and Motility”. In: *PLoS Computational Biology* 8.12 (2012). Ed. by James A. Glazier, e1002793. DOI: 10.1371/journal.pcbi.1002793.

[144] Gaudenz Danuser, Jun Allard, and Alex Mogilner. “Mathematical Modeling of Eukaryotic Cell Migration: Insights Beyond Experiments”. In: *Annual Review of Cell and Developmental Biology* 29.1 (2013), pp. 501–528. DOI: 10.1146/annurev-cellbio-101512-122308.

[145] Andreas Buttenschön and Leah Edelstein-Keshet. “Bridging from single to collective cell migration: A review of models and links to experiments”. In: *PLOS Computational Biology* 16.12 (2020). Ed. by Alex Mogilner, e1008411. DOI: 10.1371/journal.pcbi.1008411.

[146] Baichuan Yuan, Hao Li, Andrea L. Bertozzi, P. Jeffrey Brantingham, and Mason A. Porter. “Multivariate Spatiotemporal Hawkes Processes and Network Reconstruction”. In: *SIAM Journal on Mathematics of Data Science* 1.2 (2019), pp. 356–382. DOI: 10.1137/18M1226993.

[147] Richard H. Kessin, Adam Kuspa, William F. Loomis, et al. *Dictyostelium discoideum Protocols*. Ed. by Ludwig Eichinger and Francisco Rivero. 1st ed. Vol. 346. Methods in Molecular Biology. Totowa, NJ: Humana Press, 2006, pp. 1–564. DOI: 10.1385/1597451444.

[148] Maike Stange, Ted Moldenhawer, and Carsten Beta. *Fluorescent (C)LSM image sequences of Dictyostelium discoideum (Ax2 - LifeAct mRFP) for cell track and cell contour analysis*. Dryad, Dataset. 2020. DOI: 10.5061/dryad.b5mkkwhbd.

[149] Laurent Golé, Charlotte Rivière, Yoshinori Hayakawa, and Jean-Paul Rieu. “A Quorum-Sensing Factor in Vegetative Dictyostelium Discoideum Cells Revealed by Quantitative Migration Analysis”. In: *PLoS ONE* 6.11 (2011). Ed. by Joseph Najbauer, e26901. DOI: 10.1371/journal.pone.0026901.

[150] Luke Tweedy, Börn Meier, Jürgen Stephan, Doris Heinrich, and Robert G. Endres. “Distinct cell shapes determine accurate chemotaxis”. In: *Scientific Reports* 3.1 (2013), p. 2606. DOI: 10.1038/srep02606.

[151] Jakob Löber, Falko Ziebert, and Igor S. Aranson. “Collisions of deformable cells lead to collective migration”. In: *Scientific Reports* 5.1 (2015), p. 9172. DOI: 10.1038/srep09172.

[152] Sven Flemming, Francesc Font, Sergio Alonso, and Carsten Beta. “How cortical waves drive fission of motile cells”. In: *Proceedings of the National Academy of Sciences* 117.12 (2020), pp. 6330–6338. DOI: 10.1073/pnas.1912428117.

[153] Eduardo Moreno, Robert Großmann, Carsten Beta, and Sergio Alonso. “From Single to Collective Motion of Social Amoebae: A Computational Study of Interacting Cells”. In: *Frontiers in Physics* 9.February (2022), pp. 1–17. DOI: 10.3389/fphy.2021.750187.

[154] James H. Stagge, David E. Rosenberg, Adel M. Abdallah, et al. “Assessing data availability and research reproducibility in hydrology and water resources”. In: *Scientific Data* 6.1 (2019), p. 190030. DOI: 10.1038/sdata.2019.30.

[155] Jason A. Papin, Feilim Mac Gabhann, Herbert M. Sauro, David Nickerson, and Anand Rampadarath. “Improving reproducibility in computational biology research”. In: *PLOS Computational Biology* 16.5 (2020), e1007881. DOI: 10.1371/journal.pcbi.1007881.

[156] Christian M. Stracke. “Open Science and Radical Solutions for Diversity, Equity and Quality in Research: A Literature Review of Different Research Schools, Philosophies and Frameworks and Their Potential Impact on Science and Education”. In: *Radical Solutions and Open Science: An Open Approach to Boost Higher Education*. Lecture Notes in Educational Technology. Springer Singapore, 2020, pp. 17–37. DOI: 10.1007/978-981-15-4276-3_2.

[157] Benedikt Fecher and Sascha Friesike. *Open Science: One Term, Five Schools of Thought*. Ed. by Sönke Bartling and Sascha Friesike. 1st ed. Cham: Springer International Publishing, 2014. Chap. 2, pp. 17–47. DOI: 10.1007/978-3-319-00026-8_2.

[158] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, et al. “The FAIR Guiding Principles for scientific data management and stewardship”. In: *Scientific Data* 3.1 (2016), p. 160018. DOI: 10.1038/sdata.2016.18.

[159] *Artifact Review and Badging Version 1.1*. 2020. URL: https://www.acm.org/publications/policies/artifact-review-and-badging-current (visited on Sept. 8, 2022).

[160] Creative Commons. *About CC Licenses*. URL: https://creativecommons.org/about/cclicenses/ (visited on Sept. 8, 2022).

[161] Open Source Initiative. *GNU General Public License version 3*. URL: https://opensource.org/licenses/GPL-3.0 (visited on Sept. 8, 2022).

[162] Open Source Initiative. *The MIT License*. URL: https://opensource.org/licenses/MIT (visited on Sept. 8, 2022).

[163] Open Source Initiative. *Apache License, Version 2.0*. URL: https://opensource.org/licenses/Apache-2.0 (visited on Sept. 8, 2022).

[164] Open Source Initiative. *2-Clause BSD License*. URL: https://opensource.org/licenses/BSD-2-Clause (visited on Sept. 8, 2022).

[165] Deutsche Forschungsgemeinschaft (DFG). *Guidelines for Safeguarding Good Research Practice. Code of Conduct*. Tech. rep. Deutsche Forschungsgemeinschaft (DFG), 2022, pp. 1–26. DOI: 10.5281/zenodo.6472827.

[166] Paola Masuzzo and Lennart Martens. “An open data ecosystem for cell migration research”. In: *Trends in Cell Biology* 25.2 (2015), pp. 55–58. DOI: 10.1016/j.tcb.2014.11.005.

[167] Alejandra N. Gonzalez-Beltran, Paola Masuzzo, Christophe Ampe, et al. “Community standards for open cell migration data”. In: *GigaScience* 9.5 (2020), pp. 1–11. DOI: 10.1093/gigascience/giaa041.

[168] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, et al. “Fiji: an open-source platform for biological-image analysis”. In: *Nature Methods* 9.7 (2012), pp. 676–682. DOI: 10.1038/nmeth.2019.

[169] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. “Cellpose: a generalist algorithm for cellular segmentation”. In: *Nature Methods* 18.1 (2021), pp. 100–106. DOI: 10.1038/s41592-020-01018-x.

[170] Piotr Baniukiewicz, Sharon Collier, and Till Bretschneider. “QuimP: analyzing transmembrane signalling in highly deformable cells”. In: *Bioinformatics* 34.15 (2018). Ed. by Robert Murphy, pp. 2695–2697. DOI: 10.1093/bioinformatics/bty169.

[171] Hao Wu, Alexander Farutin, Wei-Fan Hu, et al. “Amoeboid swimming in a channel”. In: *Soft Matter* 12.36 (2016), pp. 7470–7484. DOI: 10.1039/C6SM00934D.

[172] H. G. Othmer. “Eukaryotic cell dynamics from crawlers to swimmers”. In: *WIREs Computational Molecular Science* 9.1 (2019), pp. 1–24. DOI: 10.1002/wcms.1376.

[173] F. O. Ribeiro, M. J. Gómez-Benito, J. Folgado, P. R. Fernandes, and J. M. García-Aznar. “Computational model of mesenchymal migration in 3D under chemotaxis”. In: *Computer Methods in Biomechanics and Biomedical Engineering* 20.1 (2017), pp. 59–74. DOI: 10.1080/10255842.2016.1198784.

[174] Yulia Artemenko, Thomas J. Lampert, and Peter N. Devreotes. “Moving towards a paradigm: common mechanisms of chemotactic signaling in Dictyostelium and mammalian leukocytes”. In: *Cellular and Molecular Life Sciences* 71.19 (2014), pp. 3711–3747. DOI: 10.1007/s00018-014-1638-8.

[175] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. “A Limited Memory Algorithm for Bound Constrained Optimization”. In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208. DOI: 10.1137/0916069.

[176] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. “Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization”. In: *ACM Transactions on Mathematical Software* 23.4 (1997), pp. 550–560. DOI: 10.1145/279232.279236.

[177] Marucha Lalee, Jorge Nocedal, and Todd Plantenga. "On the Implementation of an Algorithm for Large-Scale Equality Constrained Optimization". In: *SIAM Journal on Optimization* 8.3 (1998), pp. 682–706. DOI: 10.1137/S1052623493262993.

[178] Richard H. Byrd, Mary E. Hribar, and Jorge Nocedal. "An Interior Point Algorithm for Large-Scale Nonlinear Programming". In: *SIAM Journal on Optimization* 9.4 (1999), pp. 877–900. DOI: 10.1137/S1052623497325107.

[179] Alan C. Hindmarsh. *ODEPACK, a systematized collection of ODE solvers*. In: Scientific Computing. Ed. by R. S. Stepleman, M. Carver, R. Peskin, et al. Vol. 1. of IMACS Transactions on Scientific Computation. North-Holland Publishing Company, Amsterdam, 1983, pp. 55-64. URL: https://computing.llnl.gov/projects/odepack/publications.

[180] Linda Petzold. "Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations". In: *SIAM Journal on Scientific and Statistical Computing* 4.1 (1983), pp. 136–148. DOI: 10.1137/0904010.